FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Framework BigData

Maria Inês Monteiro Varandas

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Pedro João

Co-orientador externo: Jorge Amaral

24 de Julho de 2018

Resumo

Com os avanços tecnológicos houve um crescimento explosivo da quantidade de dados gerados em todo o mundo. Hoje em dia é gerada mais informação em dois dias do que em toda a civilização até 2003. Com esse crescimento surgiu necessidade de analisar a informação com o objetivo de lhe atribuir valor. Do ponto de vista empresarial, a análise das suas operações representa, cada vez mais, uma crescente necessidade para alcançar sucesso.

Dada a urgência de responder às necessidades da nossa sociedade, é imperativo a existência de sistemas de análise de dados em tempo real que para produzir resultados atinjam níveis de latência muito baixos, como segundos ou até milissegundos.

Como tal, o objetivo da dissertação descrita neste documento reside na construção e implementação de uma *framework* de monitorização para projetos de *Big Data* implementados em ambiente *Microsoft Azure*. Foram realizadas monitorizações ao nível da qualidade de dados, respostas de APIs externas e *logs* de diagnóstico. Estas, proporcionam ao utilizador, controlo e perceção dos acontecimentos do processo em questão através de relatórios no *PowerBI*, atualizados em tempo real. Este projeto foi motivado por uma necessidade da empresa BUSINESSTOFUTURE que, desta forma, potenciou a implementação desta *framework*.

Por último, foram feitos testes computacionais enviando mensagens para o processo com o intuito de perceber a veracidade, consistência e rapidez da informação visualizada no *Power BI* e das respetivas métricas. Foi também realizada uma análise orçamental sobre os recursos do *Microsoft Azure* usados na construção da *framework de monitorização*.

Abstract

With technological advances there has been an explosive growth in the amount of data generated around the world. Nowadays, more information is generated in two days than in all of civilization until 2003. With this growth came the need to analyze information to give it value. From the business point of view, the analysis of its operations represents an increasing need to achieve success.

Due to the urgency of responding to the needs of our society it is imperative to have real-time data analysis systems that produce very low latency levels, such as seconds or even milliseconds, to produce results.

The purpose of this dissertation is the implementation of a monitoring framework for Big Data projects implemented in a Microsoft Azure environment. Data quality, external API responses, and diagnostic logs were monitored, providing the user control and perception of the process events through reports in PowerBI, updated in real time. This project was motivated by the need of the BUSINESSTOFUTURE company to implement this framework.

In the end, computational tests were made by sending messages to the process in order to understand the veracity, consistency and speed of the information visualized in Power BI and its metrics. A budget analysis was also carried out on the Microsoft Azure resources used in the construction of the monitoring framework.

Agradecimentos

Em primeiro lugar, queria agradecer ao meu orientador Pedro João pela orientação prestada, pela disponibilidade e pelo apoio demonstrado.

À B2F que me possibilitou a realização desta dissertação e que esteve sempre disponível para me ajudar em qualquer dificuldade que encontrasse pelo caminho. Obrigada equipa de BI pela amizade, ajuda e paciência que tiverem comigo. Um especial obrigada ao meu amigo Francisco Capa que foi incansável comigo em todos os momentos deste percurso. Pelos incentivos, pela amizade, pela compreensão e paciência que sempre demonstrou ao longo da realização desta dissertação.

Um agradecimento especial ao Francisco Pinto, por estar sempre ao meu lado, por ser sempre o meu apoio incondicional, por me corrigir quando estou errada, por me fazer sempre acreditar que é possível ser melhor e mais forte e principalmente pela paciência e motivação nos momentos menos bons.

Quero agradecer a todos amigos que fiz ao longo deste percurso na FEUP, todos foram essenciais para o meu crescimento pessoal e profissional. Em especial à Patrícia Saraiva, que foi sempre a minha companheira e amiga, esteve sempre pronta para ajudar, motivar e para ouvir os meus desabafos. Às minhas meninas Bárbara Coelho e Leonor Mendes Freitas que que me marcaram de uma forma muito especial.

À Carla, a amiga de todas as horas, pela amizade que fomos criando ao longo deste percurso, pela motivação, pelo companheirismo, por toda a amizade, conselhos e brincadeiras.

À Manuela, que esteve sempre presente, mesmo com todos os obstáculos. Obrigada por seres sempre um apoio incondicional na minha vida.

Ao Tetecagas, que de forma direta ou indireta, estiveram sempre presentes em todas etapas deste percurso com amizade e diversão.

E por fim, aos mais importantes, à minha família que são o meu exemplo de vida. À minha mãe por dar os melhores conselhos do mundo, pela amizade, pelo apoio, pelos raspanetes ao longo deste percurso que me tornaram sem dúvida na pessoa que hoje sou. Ao meu pai, pelo seu sorriso, pois mesmo nos piores momentos este está sempre presente. Obrigada por me ensinares a não desistir e a lutar sempre pelos meus sonhos. À minha avó, que estaria super orgulhosa da sua menina que ensinou a ser uma mulher de armas. A vocês, dedico esta dissertação.

Inês Varandas

"I believe that empowered women change society.
The data tells us."

Melinda Gates,
Cofounder of the Bill & Melinda Gates Foundation

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AD	Armazenamento de Dados
AMQP	*Advanced Message Queuing Protocol*
AWS	*Amazon Web Services*
BI	*Business Intelligence*
B2F	*BUSINESSTOFUTURE*
d	Dias
CR	Consumo do Recurso
DM	*Data Mart*
DW	*Data Warehouse*
DSA	*Data Staging Area*
e	Eventos
EE	Eventos de entrada
EX	Execuções
GB	*Gigabyte*
GCP	*Google Cloud Platform*
h	Horas
HTTP	*HyperText Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
IaaS	*Infrastructure as a Service*
IoT	*Internet-of-things*
IT	*Information Tecnology*
JSON	*JavaScript Object Notation*
m	Mês
M	Milhões
MB	*Megabyte*
OE	Operações de escrita
OTD	Operações e transferências de dados
PaaS	*Platform as a Service*
s	Segundos
SaaS	*Software as a Service*
SOAP	*Simple Object Access Protocol*
SSL	*Secure Sockets Layer*
SU	*Streaming Units*
TB	*Terabyte*
TLS	*Transport Layer Security*
UD	Unidades de débito
XML	*Extensible Markup Language*

Capítulo 1

Introdução

O presente documento apresenta o projeto no âmbito da dissertação de Mestrado Integrado em Engenharia Eletrotécnica e de Computadores realizada em ambiente empresarial na BUSINESTOFUTURE (B2F). Neste capítulo introdutório será explicado em que contexto o projeto foi desenvolvido, a descrição do problema, os respetivos objetivos e a estrutura do documento.

1.1 Enquadramento e motivação

Nos últimos anos tem existido um aumento exponencial da tecnologia utilizada no mundo. É estimado que 90 % do total de dados do mundo foi gerado nos últimos dois anos [10], vivendo-se, portanto, num mundo onde tudo é digital e consequentemente tudo gera informação. A informação está presente no quotidiano das populações, mesmo que não seja percetível. Desde a informação de quanta água é gasta por mês, até quais as séries mais vistas no último mês por uma pessoa específica.

Com esse crescimento surgiu necessidade de analisar a informação com o objetivo de lhe atribuir valor. Do ponto de vista empresarial, a análise das suas operações representa, cada vez mais, uma crescente necessidade para alcançar sucesso. Consequentemente, os sistemas tradicionais usados para fazer análise da informação tornaram-se obsoletos e sentiu-se necessidade de recriar os sistemas antigos. Assim, surgiram conceitos como *Big Data* e *Cloud Computing*. Estes, procuram resolver falhas nos sistemas tradicionais e aumentar a eficiência e as possibilidades dos processos modernos.

Dada a relevância dos conceitos de *Big Data* e *Cloud Computing*, esta dissertação tem como objetivo a implementação de uma *framework* de monitorização de projetos de *Big Data* e também a partilha de conhecimentos nesta área específica.

1.2 Objetivos

Esta dissertação de mestrado em ambiente empresarial foi proposta pela empresa BUSINESTOFUTURE, fundada em 2006 e que conta com cerca de 30 colaboradores. A sua atividade

focaliza-se no desenho e desenvolvimento de soluções empresariais de gestão de informação e sistemas de apoio à decisão para médias e grandes empresas, num conjunto alargado de mercados. Ou seja, a B2F é especialista em projetos de *Business Inteligence* (BI) onde oferece as melhores práticas no desenvolvimento deste tipo de soluções usando tecnologias de ponta. Além de um desenvolvimento à medida centrado no cliente, partilha conhecimento e experiências com as empresas que trabalha com o objetivo de colmatar necessidades mais exigentes.

Os objetivos propostos pela B2F no âmbito desta dissertação foram desenhar e implementar uma *framework* de monitorização para arquiteturas *Big Data* que fosse capaz de:

- Ingerir informação em tempo real de várias fontes (API, webhooks, IOT);
- A criação de *dashboards* para visualizar as monitorizações;
- Acionar alertas/ações de vários tipos;

1.3 Descrição do problema

1.3.1 *Framework* de monitorização

Os processos que envolvem sistemas financeiros, onde é usada informação pessoal dos clientes, são considerados processos críticos, o que significa que não podem existir falhas. Uma falha neste tipo de processos pode significar uma perda significativa para a empresa tanto na parte financeira da mesma, como na confiança cliente-empresa. Assim sendo, é necessário garantir que o processo corra como planeado e que não haja erros que passam despercebidos.

Com esse intuito, o problema proposto pela empresa no âmbito desta dissertação, consiste em desenhar, construir e implementar uma *framework* de monitorização para um sistema de Big Data com transações de vários tipos em tempo real. Ou seja, um processo que se enquadre em todos os casos, que possa ser escalável e flexível para abranger todas as necessidades.

Com o objetivo de criar uma *framework* que preencha os requisitos da empresa, foi implementado o caso de estudo, referente ao sistema de alerta de fraudes, com o objetivo de detetar erros e casos que possam ser relevantes monitorizar.

1.3.2 Caso de estudo de sistema de alerta de fraudes

Surgiu por parte de um cliente da B2F a necessidade da criação de um sistema de alerta de fraudes que alerte o cliente em caso de bloqueio do cartão multibanco. Desta forma, o utilizador do cartão tem informação sobre transações fraudulentas e até mesmo as razões pelas quais o seu cartão foi bloqueado, tudo em tempo real.

Isto é vantajoso tanto para a empresa como para o cliente. Do lado da empresa financeira é benéfico pois há conhecimento, em tempo real, de todas as transações efetuadas detalhadamente, incluindo a data da transação, o cliente, a localização, a razão do bloqueio do cartão, entre outras. Isto faz com que a empresa tome decisões rápidas e eficazes sobre as transações realizadas, tanto

a nível de negócio, como marketing ou até mesmo para a definição de estratégias. Da parte do cliente, este recebe uma mensagem pelo telemóvel no caso de bloqueio do cartão juntamente com a causa do bloqueio.

Esta informação ajuda o cliente a ter uma reação mais eficaz e ponderada sobre os movimentos do seu cartão. Por exemplo, na situação hipotética de um cliente viajar para três sítios diferentes no mesmo dia e usar o seu cartão em cada um deles, a reação normal de uma instituição financeira é bloquear o cartão, chamado o falso-positivo. Nesta situação seria muito vantajoso um processo que alertasse o cliente no caso do seu cartão multibanco ser bloqueado pois o cliente poderia ligar imediatamente para o banco para este corrigir o erro.

1.4 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos. No capítulo 2, é descrito o estado da arte onde é realizado um estudo referente aos temas essenciais para a consumação desta dissertação. No capítulo 3, é abordada a solução proposta para a *framework* de monitorização. Neste capítulo é referenciado também o objetivo e o funcionamento de cada recurso do *Microsoft Azure* usado na implementação deste projeto. No capítulo 4 é demonstrada a implementação da *framework* de monitorização no ambiente *Microsoft Azure*. No capítulo 5 são apresentados os resultados do projeto em questão. Neste ponto são apresentados os relatórios realizados no PowerBI e também uma análise orçamental do mesmo. No capítulo 6, são apresentadas as ideias e conclusões adquiridas durante o desenvolvimento do projecto.

Capítulo 2

Revisão Bibliográfica

2.1 Conceito de Dados

Segundo Nathan Marz, "A data system answer questions based on information that was acquired in the past" [11]. Um perfil de uma rede social responde a questões como: "Qual é o nome da pessoa?", "Quantos amigos esta pessoa tem?" tal como uma página web de um banco responde a questões como "Qual é o meu saldo atual?", " Quantas transações ocorreram na minha conta recentemente?". Com intuito de dar uma resposta fidedigna a essas questões é essencial que exista informação sobre as mesmas. Assim surgiu a importância de armazenamento de dados. Os sistemas de dados combinam dados para produzir respostas. Por exemplo, uma conta bancária é baseada na combinação de toda a informação sobre as transações nessa mesma conta. Outra observação essencial para perceber o conceito de dados, é que a informação contida em bits não é toda igual, algumas informações são derivadas de outras peças de informações. O saldo bancário é um exemplo perfeito, é evidente que o saldo da conta é derivado do histórico de transações. Quando se rastreia a origem da informação chegamos a informação "crua", esta pode ser definida por informação que não tem derivação. A esta informação chama-se *data*. [11]

2.2 Business Inteligence

Citando Hans Peter Luhn em 1958 "Comunicação eficiente é a chave para o progresso do esforço humano em todas as áreas". Os métodos existentes de análise de dados existentes naquela altura tornaram-se desadequados comparativamente com o exponencial crescimento da produção de dados que existe atualmente [10]. Assim, Howard Dresner do Gartner Group em 1989 introduziu o termo *Business Inteligence* (BI) para sistemas de apoio à decisão, sistemas de informação executivos e de gestão da informação [12].

Hoje em dia, as empresas têm necessidade de saber o que está a acontecer no momento, o que é previsível que aconteça e que medidas tomar para obter um melhor desempenho. Por essa razão, os sistemas de BI são ferramentas de trabalho essenciais no tratamento de dados de uma empresa e consequentemente essenciais no aumento da sua performance [12], pois permitem o apoio a

decisões estratégicas através de uma constante recolha de dados, organização, análise, partilha e acompanhamento desses mesmos dados [13]. O conceito de BI é abrangente, porém pode ser definido como uma combinação de produtos, tecnologias e metodologias que organizam informação chave que uma empresa necessita para melhorar os lucros e a sua performance. É considerado como a relação entre a gestão e a tecnologia, onde a matéria prima é a informação e o seu produto final o conhecimento devido da mesma [14]. Este conceito tornou então possível a análise e visualização de dados anteriormente considerados como desperdício e agora considerados como bens essenciais para o progresso e tomada de decisões estratégicas de uma empresa [10].

Os sistemas de *Business Intelligence* têm várias responsabilidades e objetivos. Devem garantir um ambiente de dados fiável e simples para que a sua gestão seja mais eficaz. Vai consequentemente aumentar a transparência e a compreensão do negócio e disponibilizar um conhecimento em tempo real permitindo assim aos gestores ter uma perspetiva clara das áreas que devem controlar. Sustentam as decisões tomadas pela empresa, permitindo decisões eficazes [14].

Para que uma empresa rentabilize ao máximo o investimento em sistemas de BI devem ser estabelecidas as necessidades da empresa, o seu objetivo específico e em que mercado se insere [15].Os dados são tratados e comparados com eventos anteriores ou com dados externos melhorando dessa forma a disponibilidade e qualidade da informação para a tomada de decisão [16].

A figura 2.1 representa a arquitetura de um sistema BI. Os dados são recolhidos de várias fontes de dados, posteriormente, através dos processos como a Extração, Transformação e Carregamento (ETL) que vai ser abordado no capítulo 2.4. De seguida os dados são então carregados e armazenados num *Datawarehouse* e vão ser geridos por um ou mais servidores. Os DWs são complementados por um conjuntos de servidores de nível intermédio que fornecem funcionalidades especializadas para diferentes cenários de BI.

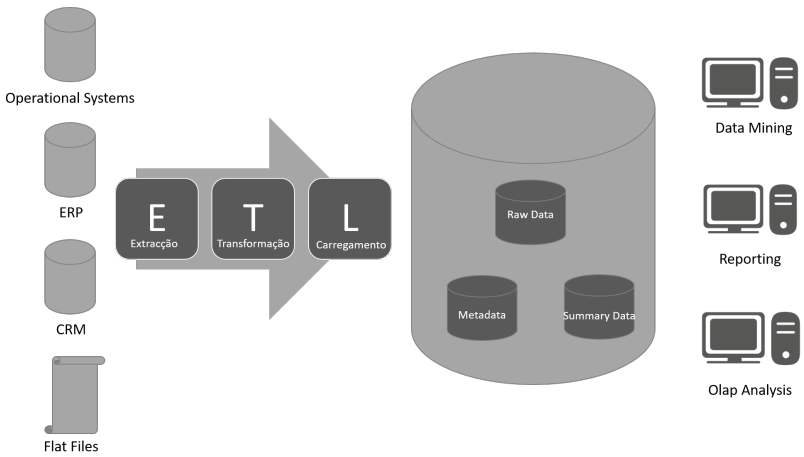

Figura 2.1: Arquitetura de um sistema de BI (Elaboração própria)

2.3 *Data Warehouse*

Quando se pensa na origem de um *Data Warehouse* inevitavelmente falamos de Raplh Kimball e William Bill Inmon. Cada um com o seu peso na área. O primeiro conhecido como 'Pai do BI' e o segundo como 'Pai do DW'. Assim, segundo William Bill Inmon [17], o *Data Warehouse* é uma parte integrante de um sistema de *Business Intelligence*. É, como a própria definição indica, um armazém de dados onde estes são recolhidos de sistemas transacionais.

Este permite a análise de grandes quantidades de dados, possibilitando uma análise mais precisa de eventos passados, uma tomada de decisão mais consciente e a previsão de eventos futuros [16]. No ambiente de DW os utilizadores conseguem aceder a informação utilizando ferramentas simples e de fácil utilização, em vez de depender de relatórios fornecidos por especialistas de IT. Tornando assim, as empresas mais independentes e confiantes nas suas decisões [18]. É, de uma forma simplificada, um repositório de dados centralizado. O DW é orientado ao assunto, integrado, não volátil e variante no tempo [17].

- **Orientado ao assunto**

  O DW pode ser usada na análise particular de uma área.

- **Integrada**

  Integra informação de múltiplas fontes. Assim sendo mesmo que fontes diferentes tenham maneiras diferentes de identificar um produto, no DW só vai existir uma maneira de identificar um produto. A partir do momento que faz parte do DW só tem uma imagem física.

- **Não volátil**

  A informação armazenada no DW jamais é alterada.

- **Variante no tempo**

  Enquanto nos sistemas tradicionais só era guardada a informação mais recente, no DW é guardada toda a informação ao longo do tempo. Por exemplo, num sistema transacional só guardaria a informação da morada mais recente de um utilizador, no DW são guardadas todas as moradas que esse utilizador já teve.

Em fases iniciais, enquanto Inmon [17] aposta na criação de um DW que cubra todas as áreas de uma organização, Kimball [16] aponta para a criação de *Data Marts* (DM). Um *Data Mart* é uma subsecção de um *Data Warehouse* e é usualmente orientado a um negócio específico. Partindo de um repositório central normalizado, Inmon [17] sugere que sejam adicionados DM de forma a suprimir as necessidades de cada departamento. Por sua vez, Kimball [16] propôs que os diferentes DM modelados, usando dimensões sejam virtualmente integrados. Estas duas abordagens são normalmente designadas respetivamente de top-down e bottom-up. Em conclusão, eles abordam o problema com diferentes filosofias, técnicas de modelação e estratégias de implementação.

2.3.1 Tabelas de Facto e de Dimensão

Antes de se definir um modelo de dimensão para o *Data Mart*, é importante perceber o conceito de tabelas de factos e de dimensão. As tabelas de factos guardam informação das medidas de performance da empresa relativas ao negócio em questão [16]. Cada linha da tabela representa um evento associado a um processo que contém as informações de medida associadas a esse mesmo evento. A informação é usualmente numérica e é facilmente manipulada para métricas. Em suma, os factos contêm a informação que realmente tem significado para a análise e consequente desenvolvimento da empresa.

As dimensões descrevem os objetos envolvidos num projeto de BI. Enquanto os factos correspondem a eventos, as dimensões correspondem a pessoas, produtos ou outros objetos. As tabelas de factos contêm detalhes sobre cada instância de um objeto [19]. Estas descrevem quem, o quê, onde, quando, como e o porquê associado ao evento. As dimensões tornam a informação dos relatórios percetível e clara [16].

2.3.2 Modelação Dimensional

Segundo Kimball [16], um ambiente de um *Datawarehouse* é completamente diferente de um ambiente operacional e consequentemente as técnicas utilizadas para modelar bases de dados operacionais, não são apropriadas para modelar DW. Por essa razão, Kimball propôs uma nova técnica de modelação de dados específica para DW conhecida por modelação dimensional. Técnica esta que dispõe a informação de uma forma intuitiva e que permite um acesso mais rápido e eficaz [20].

Usando uma modelação dimensional, existem então vários esquemas sendo os mais comuns o *Star Schema* e o *Snowflake Schema*. Como o próprio nome indica, o *Star Schema* é um esquema em forma de estrela onde cada tabela de dimensão está ligada à tabela de factos que se encontra num lugar central, como se pode verificar na figura 2.3. No caso do *Snowflake Schema*, as tabelas de dimensão são relacionadas com outras tabelas de dimensão com o objetivo de normalizar a informação das dimensões, figura 2.2. [16].

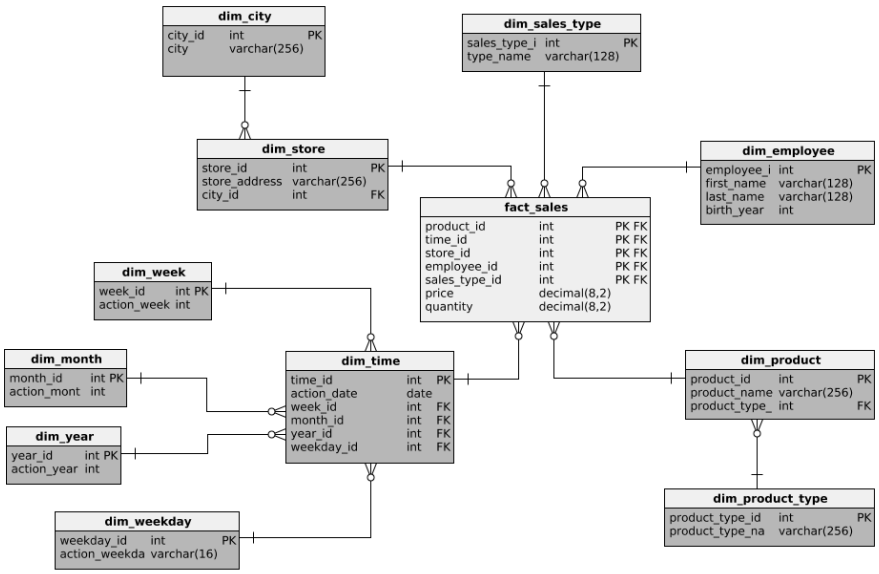

Figura 2.2: Esquema *Snowflake* (Elaboração própria)

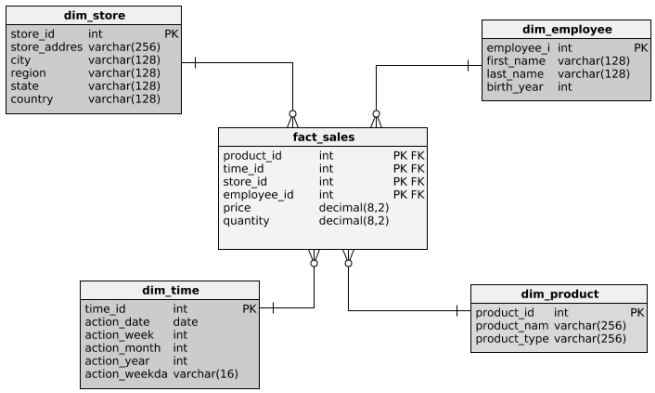

Figura 2.3: Esquema Star Schema (Elaboração própria)

Estes dois esquemas têm características e propósitos diferentes como podemos verificar na tabela 2.1.

Tabela 2.1: Comparação entre *Star Schema* e *Snowflake Schema*

	Star Schema	***Snowflake Schema***
Acessibilidade	Facilidade de acesso à informação por parte dos analistas de negócio.	Maior dificuldade por parte dos analistas de negócio devido à elevada quantidade de tabelas de dimensão.
Tabela de Dimensão	Só existe uma tabela de dimensão para cada dimensão.	Existem várias tabelas de dimensão para a mesma dimensão com o objetivo de normalizar cada tabela de dimensão.
Complexidade da *query*	*Query* simples e de fácil utilização.	*Query* com um nível de complexidade mais elevado devido a junção de chaves estrangeiras entre tabelas de dimensão.
Performance da *query*	Alta Performance.	*Query* com tempos de execução mais elevados.
Quando Usar	Quando as tabelas de dimensão guardam um número de linhas relativamente pequeno.	Quando as tabelas de dimensão guardam um número grande de linhas.
Datawarehouse	Alta performance em qualquer tipo de DW/DM.	Melhor performance para DW/DM pequenos.

2.4 ETL- *Extract, Transform, Load*

Extração, Transformação e Carregamento (*Extract, Transform, Load* - ETL) são procedimentos de uma técnica de carregamento de dados para um *Data Warehouse* que são responsáveis pela extração de dados de várias fontes, a sua limpeza, otimização e inserção desses dados num DW [21].
São ferramentas que reduzem o tempo de desenvolvimento, gerem o fluxo de informação na cadeia de BI e providenciam soluções para gerir as alterações de informação ao longo do tempo [15]. Esta fase caracteriza-se por englobar procedimentos de limpeza, integração e transformação de dados. Este é o processo mais crítico e demorado na construção de um DW [21].

A primeira questão a tratar é quais as necessidades de um cliente de BI, é necessário perceber-se quais são os dados que é necessário de extrair para um *Data Warehouse* [22]. A extração, como a

sua designação indica, é o primeiro processo do ETL, onde é extraída a informação necessária das bases de dados disponíveis para a *data staging área* (DSA). Este é um espaço de armazenamento intermediário usada no processo de ETL entre as fontes de dados e o repositório de destino [16]. Seguidamente dá-se a transformação que é responsável por manipular a informação disponível de acordo com os objetivos de negócio. Tarefas como conversão do tipo de dados e limpeza de informação são exemplos usados no processo de transformação [23]. Por fim, é necessário carregar e agregar esta informação anteriormente cuidada para o DW onde entra então o processo carregamento [19].

2.5 Requerimentos de um *Business Inteligence/ Data Warehouse*

Segundo Kimball existem alguns requerimentos que um sistema *Business Inteligence/ Data Warehouse* precisa de garantir, tais como [16]:

- **Um sistema DW/BI deve ter** informação de fácil acesso. Ou seja, o conteúdo de um DW/BI deve ser percetível, contendo informação intuitiva e óbvia para o utilizador de negócio não apenas para o que a desenvolve. As ferramentas usadas neste tipo de sistemas devem ser simples e de fácil acesso. Devem também ter resultados no menor espaço de tempo possível.
- **Um sistema DW/BI deve ter** informação consistente. A informação presente no *Datawarehouse* deve ser credível. Com esse intuito, a informação deve ser analisada, limpa e devidamente rotulada para ter qualidade, pois a sua informação é proveniente de várias fontes.
- **Um sistema DW/BI deve ter** capacidade de mudança. Em qualquer momento do projeto pode existir a necessidade de mudanças, seja por necessidades dos utilizadores, condições de negócio ou até mesmo mudanças de tecnologias. Por esta razão, o sistema deve ser implementado de maneira a responder positivamente a qualquer tipo de mudança. A única coisa que não deve sofrer mudanças são os dados já presentes no DW pois assim podemos por em a consistência dos dados.
- **Um sistema DW/ BI deve apresentar** informação segundo uma escala temporal. Como este sistema é essencialmente usado para tomada de decisões, deve ser apresentado de uma forma temporal para que as suas decisões sejam mais conscientes e eficientes.
- **Um sistema de DW/BI deve ter** um sistema de segurança que protege a informação. Toda a informação da empresa pode ser encontrada *Data Warehouse*. Por esta razão é preciso garantir confidencialidade dos dados do DW. Para garantir total confidencialidade um sistema DW/BI deve ter um sistema de controlo de acessos.

2.6 *Big Data*

Nos dias de hoje existem 5 biliões de telemóveis em uso, 30 biliões de conteúdo partilhado no Facebook todos os meses, no Youtube são carregados por minuto mais de 300 horas de vídeo, são

feitas mais de 2 milhões de pesquisas por minuto no Google e um carro autónomo utiliza 4000 GB de dados por dia [24]. Com os avanços tecnológicos houve um crescimento explosivo da quantidade de dados gerados em todo o mundo, é gerada mais informação em dois dias do que toda a civilização até 2003 [25]. Este fenómeno que captou a atenção da indústria moderna designa-se "*Big Data*".

Big Data pode ser definido como volumes de informação disponíveis em vários graus de complexidade gerada a diferentes velocidades [26]. Informação esta, cujo o seu tamanho ultrapassa a capacidade de recolha, armazenamento e análise das típicas bases de dados tradicionais [27]. Do ponto de vista de empresa a informação é como um bem essencial, é gerada e guardada com um tamanho a volta dos Zetabytes [28]. Zikopoulos, Eaton e Deroos [29] definem que a era do *Big Data* é resultado das mudanças que têm ocorrido no mundo, onde por meio de avanços tecnologias, foi possível que pessoas e programas comunicassem constantemente. Como nos dias de hoje nos deparamos com uma sociedade dependente de informação em todo lado e a toda a hora, foram inseridas novas tecnologias, processos e competências à informação e às pessoas por quem esta é utilizada. *Big Data* trabalha com conjuntos de dados estruturados e não estruturados [23].

Rick Smolan, considera que o *Big Data* tem potencial para ser uma "*dashboard* da humanidade" e que é uma ferramenta inteligente que ajudará a combater a pobreza, o crime e a poluição. Refere também que *Big Data* é um termo de marketing mas também um avanço na tecnologia que permite entender o mundo [30]. *Big Data* é a oportunidade de definir ideias em novos tipos de dados e seu conteúdo torna o negócio mais ágil e possibilita a tomada de decisões conscientes, que anteriormente não estavam ao alcance da empresa [24].

2.6.1 Características

O conceito de *Big Data* ainda não está bem definido, mas existem alguns pontos fulcrais onde todos os autores que o tentam definir entram em concordância. Esses pontos são um grupo de "V's" que representam um grupo de características que sustentam a infraestrutura do *Big Data* [31].

Sam Madden [32] diz "*Big Data is data that is too big, too fast or too hard*". *Too Big* diz respeito à característica do *Big Data* Volume, *too fast* refere-se à velocidade e *too hard* à variedade [32] Este grupo de características é então composto por 3 "v's": Variedade, Velocidade e Volume [29]. No entanto, alguns autores admitem que é adequado acoplar mais 2 "v's": Valor e Veracidade [31]. Portanto o *Big Data* assenta em 5 pilares, como se verifica na figura 2.4: Volume, Velocidade, Variedade, Valor e Veracidade.

Como já foi referido anteriormente, atualmente existe uma quantidade de dados armazenados em larga escala. Assim sendo, retirar valor da informação desta grandeza é um verdadeiro desafio. A verdade é que todos esses dados são úteis para a perceção do negócio dos clientes e do mercado [29]. Assim o volume é uma das características principais e importantes do *Big Data* que impõem requisitos específicos e adicionais a todas a ferramentas até aqui existentes [31].

Geralmente a informação é gerada a alta velocidade e na maior parte das vezes com necessidade de resposta instantânea, em tempo-real. Um exemplo de dados gerados em alta velocidade

são os sensores em aparelhos de monitorização [23]. Sendo por essa razão a velocidade outra das características de *Big Data*. A velocidade pode ser definida pela rapidez com que os dados chegam e são armazenados [33].

Uma outra característica é a variedade. A informação é gerada em vários formatos, pois esta pode ser proveniente de emails, redes sociais ou sensores. Não existe nenhum controlo sobre os dados que são inseridos nem sobre a sua estrutura [23]. Os dados gerados pelas empresas tornam-se complexos devido à falta de estruturação e ao facto de não seguirem o modelo relacional tradicional [29]. Segundo Krishmam [23], esta complexidade associada à variedade de formatos tem a ver com a disponibilidade de informação adequada acerca do que está contido nos dados reais recolhidos, sendo fundamental para processamento de imagens, vídeo e grandes quantidades de texto. Para que seja possível processar estes novos formatos o sistema *Big Data* requer alta escalabilidade, capacidade de processamento distribuído, capacidade de processamento de imagens, gráficos, vídeo e som. Resumidamente, a variedade representa todos os tipos de dados como parte do processo de decisão.

Para que seja garantida a fiabilidade dos dados recebidos, o *Big Data* caracteriza-se pela sua veracidade [34]. Esta qualidade é garantida pela consistência dos dados, nomeadamente, a sua origem, recolha e métodos de processamento, utilizando infraestruturas confiáveis [31].

Por último, esta tecnologia caracteriza-se pelo valor que origina. Esta é uma característica de grande importância, definindo-se pelo valor agregado que os dados recolhidos podem trazer para o processo, atividade ou análise a que se destina. Relaciona-se diretamente com a variedade e o volume [31].

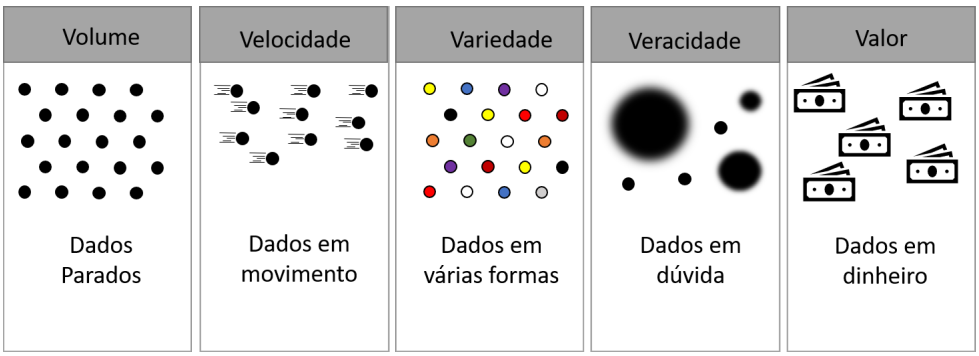

Figura 2.4: 5 V's do *Big Data* (Elaboração própria)

2.6.2 Propriedades desejadas num sistema *Big Data*

- **Robustez e tolerância a falhas**

  Os sistemas precisam de se comportar de forma correta mesmo que as máquinas parem aleatoriamente, existem semânticas complexas de consistência em bases de dados distribuídas. É imperativo para estes sistemas que sejam tolerantes a falhas humanas. Num sistema de produção é inevitável e imprevisível a ocorrência de falhas esporádicas. Da mesma forma pode acontecer uma implementação incorreta de código que pode corromper valores na base

de dados. Daí nasceu a necessidade de implementar um sistema de *Big Data* imutável e recomputável, este vai ser resistente ao erro humano pois fornece um mecanismo simples e claro de recuperação.

- **Leituras com Latência baixa e alterações**

  A maioria das aplicações de *Big Data* requer leituras com uma latência muito baixa, na ordem dos milissegundos. Por outro lado, varia conforme a aplicação específica. Umas exigem *updates* e leituras imediatas enquanto outras aplicações estão satisfeitas com latências de horas. Geralmente, quando se fala em *Big Data* é pressuposta a necessidade de leituras e *updates* imediatos. É necessário atingir níveis de latência baixos mas sem comprometer a robustez do sistema.

- **Escalabilidade**

  Escalabilidade é a habilidade de manter a performance mesmo quando há um aumento de dados. Propriedade esta essencial para os dias de hoje, onde vemos a informação a aumentar exponencialmente e sem qualquer tipo de limites.

- **Generalização**

  Um sistema de *Big Data* precisa de ser geral, pois só dessa forma é possível abranger várias aplicações. Todas as arquiteturas de *Big Data* são gerais, não são desenvolvidas para um caso específico. Tanto pode ser aplicado em sistemas de gestão de finanças, análise de redes sociais ou até aplicações científicas.

- **Extensibilidade**

  Extensibilidade é uma característica que permite adicionar uma funcionalidade com o mínimo custo de desenvolvimento. O Sistema está preparado para mudanças, facilitando assim mudanças de larga escala.

- ***Ad hoc queries***

  Como a expressão *"ad hoc"* sugere, este tipo de *query* é designado para um objetivo específico. Este tipo de *queries* contrasta com a *query* predefinida, pois esta têm o mesmo *output* em todas as suas execuções. A *ad hoc query* é criada dinamicamente e sob a necessidade do utilizador. A aptidão para fazer *ad hoc queries* é muito importante em *Big Data*. Se não se conseguir fazer perguntas aleatórias à informação não é possível descobrir o propósito da mesma.

- **Manutenção Mínima**

  A manutenção é o trabalho necessário para que o sistema corra continuamente. Com esta ação é possível antecipar quando é necessário adicionar máquinas para escalar, manter os processos em funcionamento e até fazer *debug* para aquando a existência de algum problema. Uma parte importante para minimizar a manutenção é escolher componentes que

têm a implementação mais simples possível. Em particular as bases de dados tendem a ser muito complexas internamente. Quanto mais um sistema é complexo mais difícil é perceber, sendo necessário fazer *debug* quando alguma coisa corre mal. Em *Big Data* procura-se usar sistemas simples, com algoritmos simples e componentes simples.

- ***Debuggability*** *(rastreabilidade de erros)*

  Um sistema de *Big Data* tem informação necessária para rastrear o sistema no caso de erros. É capaz de rastrear cada valor do sistema com o intuito de descobrir o que correu mal.

2.6.3 Tempo-Real

Com o avanço tecnológico e aumento da conectividade entre pessoas e dispositivos, a quantidade de dados disponíveis nas empresas, governos e organizações está em crescimento constante [35]. Em contraste com a tradicional análise de dados que recolhe periodicamente dados com grande volume e estáticos, é feita uma análise de fluxo de dados que evita ter informação parada, processando assim sempre que esta estiver disponível [36]. Devido à urgência de responder às necessidades da nossa sociedade é imperativo a existência de sistemas de análise de dados em tempo real. Sistemas estes que entre receber os dados e produzir resultados atingem níveis de latência muito baixos, como segundos ou até milissegundos [37].

O processamento de fluxo de dados permite que os dados sejam recolhidos, integrados, analisados e visualizados em tempo real, enquanto essa mesma informação está a ser produzida [38], figura 2.5. As soluções para mecanismos de fluxos de dados estão implementados com o intuito de lidar com *Big Data* em tempo real com alta escalabilidade, alta disponibilidade e alta tolerância a falhas. Estas soluções possibilitam a análise de dados em movimento, que vão responder a imensos problemas quotidianos e do futuro [39]. O objetivo do processamento em tempo real é fornecer soluções que consigam processar um fluxo de dados infinitos provenientes de várias fontes diferentes duma forma rápida e interativa.

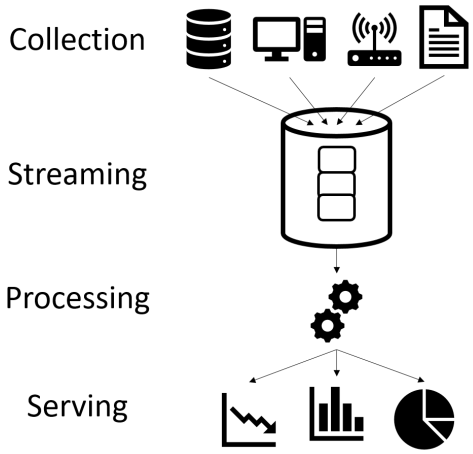

Figura 2.5: Exemplo de um sistema de *Big Data* em tempo real (Elaboração própria)

2.6.4 Arquitetura Lambda

A arquitetura Lambda atribuída a Nathan Marz [39], é uma das arquiteturas mais comuns quando se fala de processamento em tempo-real. Computar funções aleatórias em grupos de dados aleatórios em tempo real é um problema desafiante. É necessário o uso de várias ferramentas e técnicas para construir um sistema completo de *Big Data*. A ideia principal da arquitetura Lambda é construir sistemas constituídos por camadas, como exemplificado na figura 2.6. Cada camada satisfaz um grupo de propriedades e possuem relações hierárquicas entre si.

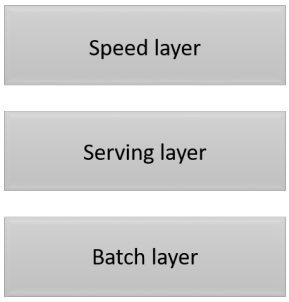

Figura 2.6: Camadas da arquitetura Lambda (Elaboração própria)

Idealmente, seria possível processar os dados sempre que fosse necessário. Com o aumento de dados, a computação deste processo iria requerer o uso de inúmeros recursos e poderia ser demasiado caro. É possível imaginar a quantidade de recursos e tempo despendido para processar um *petabyte* de informação.

A solução alternativa seria fazer uma pré computação da *query* em questão, originando assim uma *batch view*. Posteriormente, quando o utilizador quiser executar uma *query* já não vai precisar de ler toda a informação, mas sim só a *batch view*. Este processo torna o processo mais eficiente e rápido. A responsabilidade por este processo é atribuído à camada designada por camada de *Batch*. Este processo pode ser visto na imagem figura 2.7.

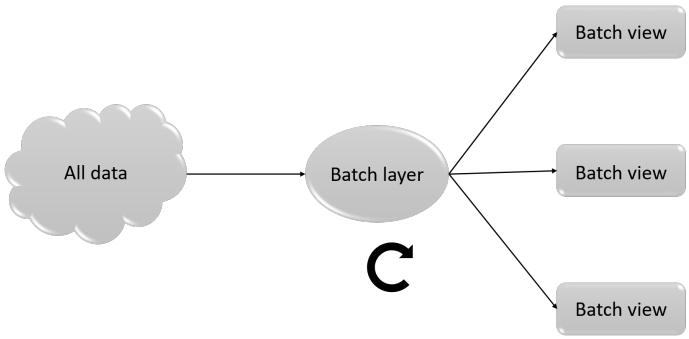

Figura 2.7: Camada *Batch* (Elaboração própria)

A camada de *Batch* tem duas funções principais: guardar conjuntos de dados imutáveis e em constante crescimento e computar funções nesse mesmo conjunto. Em bases de dados relacionais

a operação de *update* é uma das operações fundamentais. Na arquitetura lambda são criados *datasets* imutáveis que não são atualizados ou apagados, mas que em caso de necessidade podem ser acrescentados mais dados. Esta propriedade diminui o impacto da falha humana em *Big Data*. A criação de *Batch Views* é uma operação com alta latência, pois processa uma *query* em toda a informação existente. No final desta computação, vai existir informação nova, que chega no entretanto, que não vai estar incluída na *Batch view*. Assim estas vão se encontrar desatualizadas, sendo necessário a existência de outros processos.

De seguida temos uma camada denominada Camada de *Serving*. A camada de *Batch* emite *batch views* como resultado da sua função. O próximo passo é transferir esses conjuntos de dados para um sitio onde possam ser processados e analisados. Este é o ponto onde entra a camada de *Serving*. Camada esta que se baseia numa base de dados distribuída que carrega as *batch views* e permite leituras aleatórias. A base de dados da camada de *serving* suporta *updates* de *batch* e leituras aleatórias. Esta base de dados tem qualidades como robustez, facilidade de configuração, de operação e simplicidade.

Como é inteligível a camada de *serving* e de *batch* satisfazem praticamente todas as propriedade requeridas para uma sistema de *Big Data*, mencionadas no sub-Capítulo anterior. Contudo não é satisfeita uma propriedade essencial para sistemas em tempo real, alterações com baixa latência. Com o intuito de resolver essa falha, foi criado uma outra camada de *speed*.

A camada de *serving* faz alterações sempre que a camada de *batch* acaba o seu processo de computação. Isto significa que os únicos dados não representados na *batch view* são os dados que entraram enquanto a pré computação estava em execução. Portanto é necessário a existência de um sistema de dados em tempo real. Isto é, ter funções arbitrárias computadas em dados arbitrários em tempo real, só assim é possível compensar os dados inexistentes. Este é o propósito da camada de *speed*, tem como objetivo garantir que os novos dados sejam processados, analisados e representados consoante a necessidade da sua aplicação. Esta camada compensa a latência alta dos *updates* da camada de *serving*, pois aplica algoritmos rápidos e incrementais. A camada de *speed* usa bases de dados que suportam leituras e escritas aleatórias, sendo assim bases de dados muito mais complexas do que as presentes na camada de *serving*.

Ocasionalmente as camadas de *batch* e *speed* são confundidas, pois ambas produzem conjuntos de dados baseados em dados que recebem. Contudo existem grandes diferenças entre estas duas camadas. Enquanto a camada de *batch* processa toda a informação existente até ao momento, a camada de *speed* só processa os dados recentes, o que a torna muito mais rápida. Outra grande diferença é que a camada de *speed* não processa a nova informação toda de uma vez, vai computando assim que recebe *updates*, em tempo real. A camada de *batch* por outro lado faz uma computação do zero, computando toda a informação de uma vez só. As camadas *batch* e *serving* integradas apresentam exatamente o mesmo resultado final que a camada de *speed*, a única diferença está na velocidade com que os dados são processado e chegam ao destino final. Ou seja, se a aplicação que está a ser realizada não necessitar de uma resposta com latência baixa a camada de *speed* pode ser suprimida. O que pode ser vantajoso visto que esta ultima camada é muito mais complexa do que as outras duas camadas. Em conclusão, a arquitetura lambda é mostrada na

figura 2.8 e pode ser resumida nestas 3 equações:

$$\textbf{batch view} = function(alldata)$$

$$\textbf{realtime view} = function(realtimeview, newdata)$$

$$\textbf{query} = function(batchview, realtimeview)$$

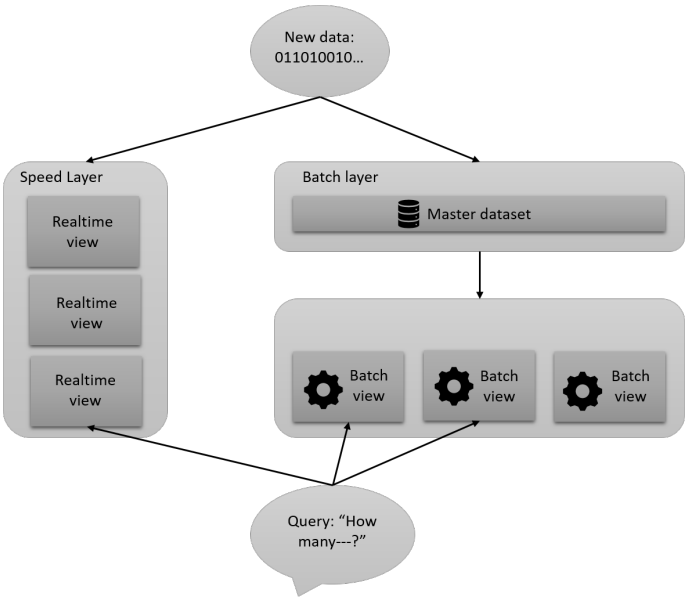

Figura 2.8: Arquitetura Lambda (Elaboração própria)

2.6.5 Arquitetura Kappa

A arquitetura Kappa foi descrita pela primeira vez pelo Jay Kreps em 2014 [40]. Uma das principais motivações para a invenção da arquitetura Kappa foi evitar a manutenção de duas bases de código separadas para a camada de *speed* e a camada de *batch*. Então, a arquitetura kappa é semelhante a arquitetura lambda mas com a camada de *batch* suprimida. Com o intuito de substituir a camada de *batch*, os dados são transmitidos apenas por um sistema de transmissão de dados. Em vez de usar uma base de dados relacional, o sistema de armazenamento de dados da arquitetura kappa é um registo imutável. A partir desse registo, os dados são transmitidos através de um sistema computacional e enviados em memórias auxiliares para a camada de *serving* [41]. Assim, nesta arquitetura temos apenas duas camadas: *Real-time* e a *Serving*. A camada de *Real-time* encadeia a informação que é enviada e os resultados das suas *queries* vão ser o *output* da camada de*Serving*, como é percebível na figura 2.9.

O objetivo é suportar um processamento de dados em tempo real e um reprocessamento contínuo num mecanismo de processamento. O Reprocessamento é um requerimento importante para que os resultados estejam atualizados. Este processo só é realizado quando algum código do trabalho de processamento de fluxo precisa ser modificado. Finalmente à semelhança da arquitetura lambda a camada de *serving* é usada para processar o resultado das *queries*.

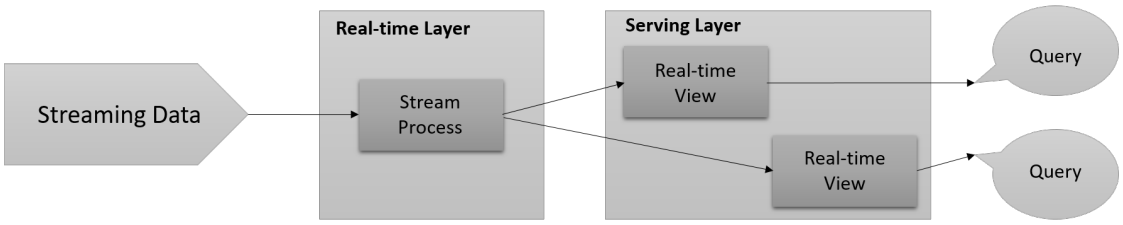

Figura 2.9: Arquitetura Kappa (Elaboração própria)

2.6.6 Kappa vs Lambda

Com o objetivo de entender as diferenças entre as arquiteturas Kappa e Lambda foi criada a tabela 2.2 com alguns critérios que são comparados entre as duas. Foi também construida a figura 2.10 que mostra as diferenças dos processos das duas arquiteturas.

Tabela 2.2: Kappa vs Lambda

Arquitetura / **Critério**	**Arquitetura Lambda**	**Arquitetura Kappa**
Arquitetura	Imutável	Imutável
Camadas	Camada *Batch*, *Serving e Speed*	Camada *Real-Time*
Processamento de dados	Camadas *Batch e Speed*	Camada *Real-Time*
Garantia de Processamento	Sim, na camada *Batch*	Uma vez com consistência
Paradigma de Reprocessamento	Em cada ciclo de *Batch*	Só quando o código é alterado
Escalabilidade	Sim	Sim
Tolerância a falhas	Sim	Sim
Armazenamento permanente	Sim	Não
Tempo-Real	Não é preciso	É preciso

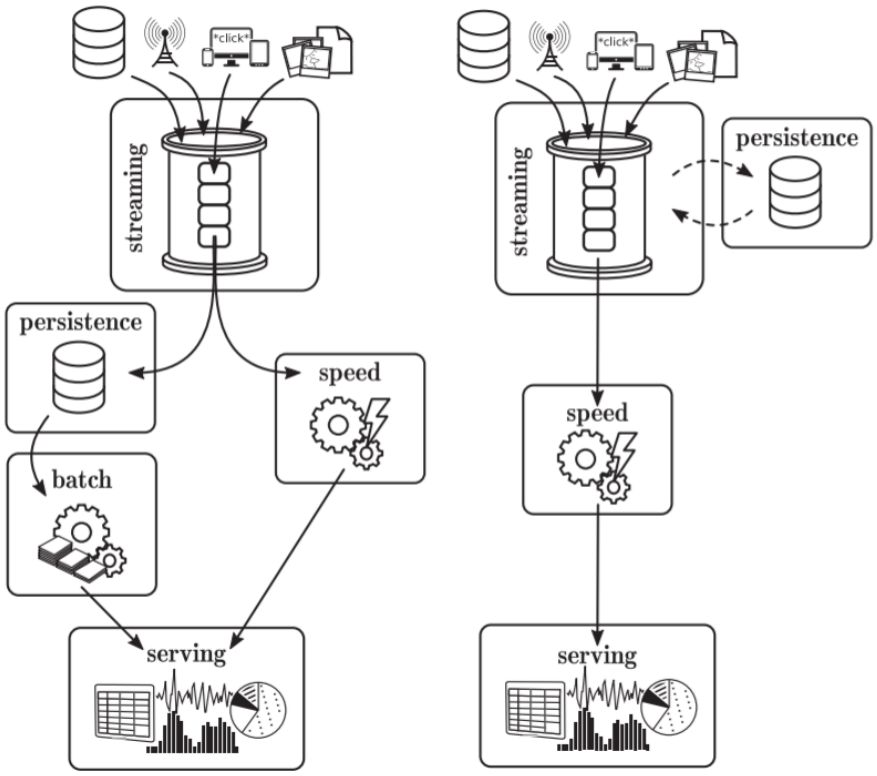

Figura 2.10: Comparação entre arquitetura Lambda e Kappa (Elaboração própria)

2.7 Computação em *Cloud*

Computação em *cloud* é uma tecnologia em crescimento que define a próxima geração de tecnologia de informação (IT) e de negócio. Com o aparecimento do *Big Data* foi sentida a necessidade de adotar serviços de computação em *cloud*, uma vez que surgiu dificuldade em obter hardware e software capaz de aguentar toda a estrutura de um sistema de *Big Data* [42]. Programadores com ideias inovadoras para serviços de Internet não necessitam, nos dias de hoje, de investimentos altos em hardware para os seus serviços e mão de obra para o operarem. Isto torna estes serviços mais acessíveis a todo o tipo de operadores [43]. Com o objetivo de atingir todo o potencial que os sistemas de *Big Data* oferecem é necessário adotar novos algoritmos de análise de dados e também novas abordagens capazes de responder às necessidades do crescimento exponencial da informação [44]. Com o uso de infraestruturas de *cloud*, são oferecidos serviços mais baratos, melhores e mais confiáveis. Esta tecnologia refere-se a aplicações oferecidas como serviços pela Internet mas também ao hardware e software dos centros de dados disponibilizados. Por definição, *Cloud* é o hardware e software de centros de dados [45].

Enquanto o conceito de *Big Data* é responsável por guardar e processar os dados, os serviços de *cloud* fornecem um ambiente de confiança, tolerante a falhas, disponível e escalável para que estes sistemas possam operar-[46]. Computação em *cloud* pode então ser definida como a

capacidade de computação infinitamente disponível e flexível. As preocupações com a largura de banda, espaço de armazenamento, poder de processamento, fiabilidade e segurança são preocupações ignoradas pelos analistas de IT que utilizam este serviços [42]. É uma tecnologia poderosa que desempenha computações de larga escala e alta complexidade. Elasticidade, pagar para usar, baixo investimento inicial e transferência de riscos são algumas das características que tornam indubitavelmente os serviços de *cloud* vantajosos e atraentes [47]. É possível agora desenvolver aplicações com um investimento inicial baixo, onde só se paga aquando o uso do recurso, com garantias de seguranças e de riscos associados. Ou seja, um serviço vantajoso para o utilizador [42]. Sistemas que suportam *Big Data* e os guardam em computação em *cloud* têm sido desenvolvidos e usados com enorme sucesso.

Na *cloud* existem 3 modelos de implementação, *Private Cloud*, *Public Cloud* e *Hybrid Cloud*. O modelo *Private Cloud* trabalha numa rede privada gerida pela empresa [48]. Este tipo de *cloud* é uma vantagem para negócios que requerem um alto nível de controlo, privacidade e segurança dos dados [49]. No caso de uma *cloud* pública a estrutura é fornecida para o uso do público em geral, trabalham numa rede publica, como a Internet [49]. Oferecem ao cliente alta eficiência e partilha de recursos a preço baixo. A privacidade, segurança e disponibilidade são características que são definidas em contrato [48]. Por fim, a *hybrid cloud* tem uma infraestrutura que consiste na composição de várias infraestruturas [48]. Combina dois modelos onde recursos adicionais de uma *cloud* pública podem ser adicionados, conforme a necessidade, à *cloud* privada[49].

2.7.1 Características

A computação em *cloud* têm essencialmente 5 características [48]:

Serviço autónomo sob demanda. O utilizador usa os serviços de *cloud* segundo a sua necessidade, podendo aumentar ou diminuir as capacidades computacionais dos recursos alocados. Capacidades como tempo de servidor ou armazenamento de rede que são alteradas sem necessidade de interação humana. Esta características oferece mais independência ao utilizador, pois usa os serviços necessários, com características definidas por ele e pelo tempo que desejar.

Por definição computação em *cloud* requer acesso à rede a partir de qualquer lugar. Assim, acesso amplo à rede significa que os serviços da *cloud* são acessíveis de qualquer plataforma. São utilizados mecanismos padrão que promovem o uso de plataformas heterogéneas. Assim, o cliente tem acesso a partir do telemóvel, PC, tablet, estações de trabalho ou qualquer outro dispositivo. É um serviço sempre disponível, sendo vantajoso para os programadores que agora não necessitam de esperar por recursos para poder desenvolver e testar as suas aplicações.

Os serviços de *cloud* oferecem recursos agrupados pois têm de atender a vários consumidores usando assim um modelo de múltiplos usuários. Os recursos computacionais da nuvem ficam reunidos geograficamente. Os seus recursos virtuais e físicos são dinamicamente atribuídos e reatribuidos pelo cliente conforme sua vontade. O cliente não controla a localização real dos recursos que está a utilizar, tendo apenas informação sobre a região em que se encontram. Os tipos de recursos são: armazenamento, processamento, memória, banda e máquinas virtuais. Até mesmo nuvens privadas tendem a reunir seus recursos entre partes da organização.

Elasticidade é definida como a capacidade de alocar mais ou menos recursos de acordo com as necessidades, com agilidade. Para o consumidor, a nuvem parece ser infinita, pois pode requisitar recursos na quantidade e hora que deseja. Ou seja a *cloud* é elástica, permite que a infraestrutura se molde às necessidades do utilizador de maneira a oferecer uma resposta rápida e eficaz.

cloud oferece também um serviço mensurável.Todos os serviços são controlados e monitorizados automaticamente pela *cloud*, com o objetivo de tornar tudo transparente tanto para o consumidor como para o fornecedor. Isso ajuda o consumidor a gerir a utilização de recursos de acordo com sua produção, e ajuda o fornecedor aquando do pagamento dos recursos.

2.7.2 Modelos de serviço

A *cloud* oferece 3 serviços diferentes: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service*(SaaS). Sistemas *Big Data* em Computação *cloud* são hospedados em *Infrastructure as a Service*, entregues como *Platform as a Service* e têm aplicações via *Software as a Service* [50]. Na figura 2.11 podemos verificar a interação dos modelos.

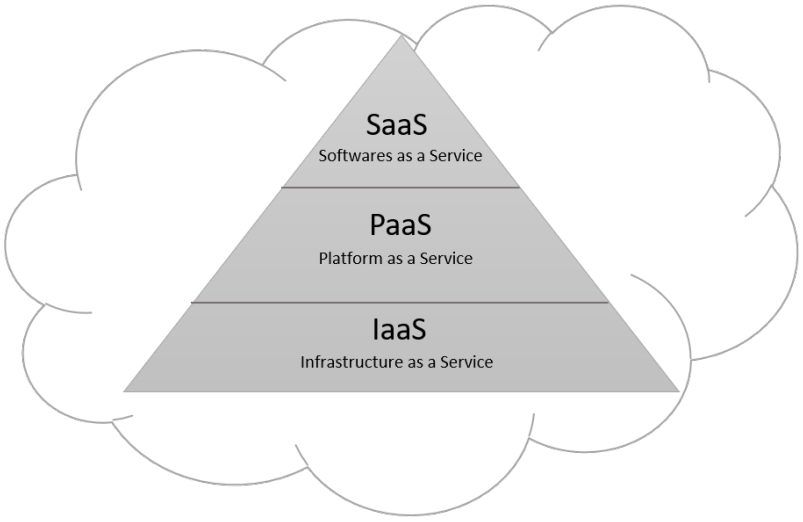

Figura 2.11: Modelos de serviço (Elaboração própria)

Infrastructure as a Service é uma infraestrutura de computação instantânea, controlada e gerida remotamente com recurso a Internet. É um serviço proveniente da *cloud* que integra toda a infraestrutura de um *datacenter* tradicional local, mas agora via Internet [46]. Ou seja, o utilizador contrata a sua infraestrutura como um serviço, alugando *datacenters* virtuais. A IaaS oferece elasticidade, pagamento por uso, *backups* automáticos, e rapidez de implementação e entrega [23]. Este serviço evita a despesa e a complexidade de obter e gerir um centro de dados. Cada recurso é oferecido como um componente distinto, pagando apenas os recursos necessários pelo tempo desejado. Os serviços de *cloud computing* gerem a infraestrutura enquanto o utilizador instala, configura e gere o seu próprio software [43], como representado na figura 2.12.

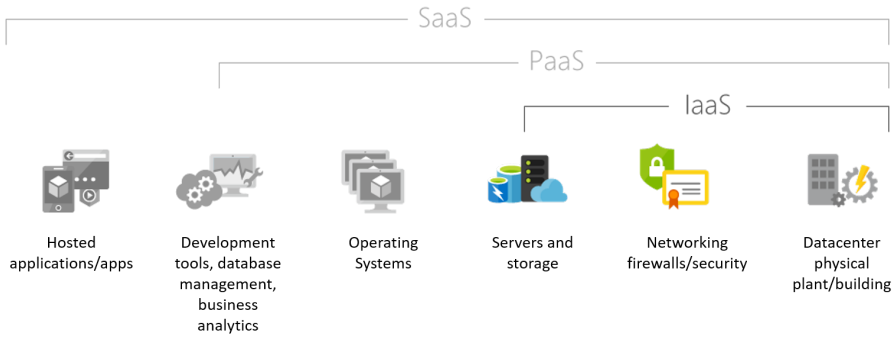

Figura 2.12: *Infrastructure as a Service* [1]

Platform as a Service é um dos modelos de *cloud computing*. É considerado um meio termo entre a IaaS e SaaS. Este modelo guarda o hardware e software necessário para desenvolver aplicações(SaaS) na própria infraestrutura [23]. A infraestrutura PaaS inclui servidores, armazenamento e *networking*, assim como o serviço IaaS, mas incluí também ferramentas de desenvolvimento e de teste, Serviços de *Business Inteligence*, gestão de sistemas de bases de dados e frameworks [43]. É oferecido a possibilidade do consumidor implementar na infraestrutura da *cloud* aplicações desenvolvidas com linguagens de programação, livrarias, serviços e ferramentas disponibilizadas pela plataforma. Ele foi criado com o intuito de suportar funções como: construir, testar, descarregar, gerir e atualizar [46], exemplificado na figura 2.13. Este modelo vai então permitir aos utilizadores decidir quais as características desejadas para construir e implementar. Um exemplo de um serviço *Platform as a Service* é o Microsoft Azure que oferece vários recursos para desenvolvimento de aplicações e contém uma infraestrutura invisível para o utilizador quando se trata de serviços de PaaS.

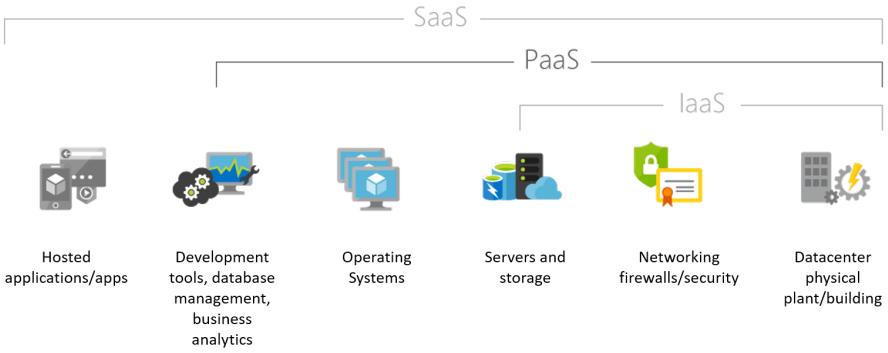

Figura 2.13: *Platform as a Service* [2]

Por ultimo, o *Software as a Service* (SaaS) permite aos utilizadores ligarem-se e utilizarem aplicações provenientes da *cloud*, através da Internet. Com esse modelo, a empresa não precisa instalar, manter e atualizar hardware ou software, como é visível na figura 2.14. O acesso é fácil e simples: apenas é necessária uma conexão com a Internet [23]. O consumidor não controla a

infraestrutura da *cloud*, as conexões à rede, servidores, sistemas operacionais, armazenamento e as capacidades da aplicações que está a usar [46]. O consumidor de um serviço de computação de *cloud* tem à sua disposição todos os recursos, que são disponibilizados na plataforma sempre que for necessário, sem precisar de qualquer interação humana. Como os serviços de *cloud* estão ligados todos em rede para aceder aos recursos só é necessário o uso de Internet. Um exemplo disso é software como *dropbox*, *google docs*, *Office 365*. São serviços de *Turn Key*, onde o utilizador não se preocupa com o hardware ou software e usa só o serviço.

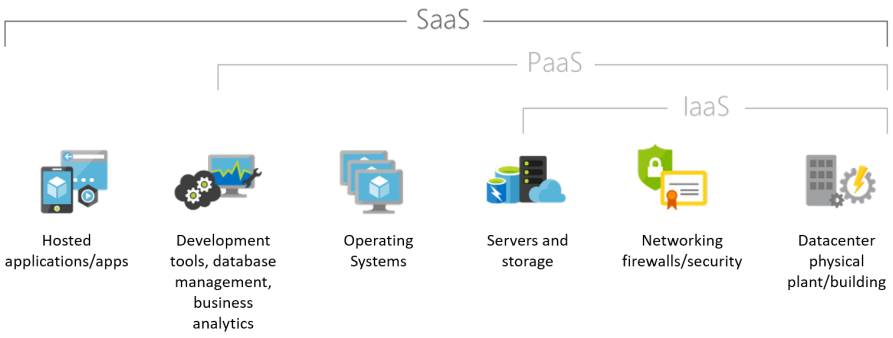

Figura 2.14: Software as a Service [3]

2.8 Tecnologias

Neste subcapítulo são apresentadas as 3 tecnologias de *Cloud Computing* mais conhecidas e competitivas. Estão são analisadas e comparadas tanto a nível de características como a nível de preço.

2.8.1 Microsoft Azure

Microsoft encontra-se em segundo em relação à Amazon segundo o uso de serviços de *cloud*. Esta infraestrutura chamada Azure foi lançada em 2010 e oferece os modelos de *Platform as a Service* e *Infrastruture as a Service* que suportam várias *frameworks* e várias linguagens de programação. Com o Azure é possível construir, implementar e gerir todos os serviços e aplicações através de uma rede global da Microsoft.

A *cloud* da Microsoft oferece vários serviços de *cloud*, como máquinas virtuais, serviços de aplicações móveis, soluções de armazenamento, sistemas de gestão de bases de dados e outros serviços.

Uma grande vantagem da Microsoft é já ter reputação entre as empresas e consequentemente pode facilmente ajudar essas mesmas empresas a fazerem a transição para serviços de *cloud*. Naturalmente, o Azure é completamente compatível com outros sistemas da Microsoft, tais como o Windows Server, System Center e Active Directory. Outro ponto forte do Azure é as capacidades como PaaS. A Azure cobra o uso de recurso por minuto enquanto a AWS cobra por hora. O que

significa que se um utilizador utilizar o recurso por 61 minutos no Azure vai pagar 61 minutos enquanto que na AWS irá pagar 2 horas.

Uma das desvantagens do Microsoft Azure é a falta de documentação e apoio técnico.

2.8.2 Google Cloud

Google oferece uma *cloud* com serviços idênticos. Esta empresa é já conhecida pelos seus serviços inovadores de *cloud computing* na comunidade de *open-source*, como é exemplo de sistemas de bases de dados em noSQL, machine learning, entre outros.

Em termos de preço é mais cara do que a Microsoft Azure mas é mais barata do que a AWS

2.8.3 AWS

Em 2006 a Amazon lançou os seus *Web Services*, o que significa que têm 12 anos de experiência em serviços de *cloud*. Igualmente, a Amazon foi o primeiro fornecedor de *cloud* a ter *Infrastructure as a Service*, permitindo que as empresas aluguem computadores virtuais. A *Amazon Web Services* (AWS) é líder quando se fala em *cloud Computing*. Para além de outras características, as suas principais são *Infrastructure as a Service*, Soluções de armazenamento, migração de conteúdo, ferramentas de desenvolvimento que o tornam mais rápido, assim como características de *machine learning*.

Tem como vantagem, em relação às outras duas *clouds* o seu início precoce comparativamente com as outras duas. É considerado um serviço com boas opções de configuração, monitorização, segurança e flexibilidade.

Uma desvantagem da AWS é ter algumas deficiências estratégias de *cloud* híbrida, ao contrário da Microsoft. Esta não considera benéfico o uso de *clouds* privadas, não apostando assim nas mesmas. Uma outra desvantagem é a quantidade de coisas que a AWS oferece. Enquanto que é uma atração em muitos casos, pode ser difícil por vezes encontrar informação sobre todos os recursos que podem ser usados.

2.8.4 *Azure vs AWS vs Google Cloud*

No mercado de *cloud computing* a seleção da melhor *cloud* é um processo difícil, pois não existe uma comparação clara sobre as possíveis ofertas de cada fornecedor. Por exemplo um fornecedor pode ser melhor na performance do CPU mas ter uma memória mais fraca ou mesmo um preço muito alto. Por essa razão, os consumidores devem escolher o serviço de *cloud* com os recursos que mais se aproxime com os seus requisitos. Escolher uma *cloud* depende então das necessidades de cada utilizador. É normal que as empresas usem vários fornecedores de serviços *cloud* para diferentes partes da operação, designando-se multi-*cloud*. Estas três empresas apresentam várias capacidades diferentes, que as distingue e ajuda o utilizador no processo de decisão.

AWS, Microsoft Azure e a Google Cloud oferecem capacidades básicas muito semelhantes, como computação flexível e armazenamento. Elas partilham os mesmos elementos de uma *cloud* pública, como por exemplo *self-service*, aprovisionamento instantâneo, *auto-scaling*, segurança e

ferramentas de gestão de identidade. Contudo estas diferem umas das outras em relação a algumas características.

Na figura 2.15 é possível observar o quadrante mágico de Gartner que diz respeito às *clouds* que oferecem *Infrastructure as a Service*, referente a 2017. É possível constatar que o serviço da Amazon é considerado líder seguido pela Microsoft.

Figura 2.15: Gartner's Magic Quadrant for Cloud Infrastructure as a Service [4]

Para ser possível comparar os serviços de *cloud* fornecidos pela AWS, Google Cloud e Microsoft Azure foi realizada uma pesquisa de recursos que cada um oferece. Foram analisados recursos nas seguintes áreas de análise: computação, armazenamento, *networking*, Bases de Dados e Segurança. Desta forma é possível comparar os 3 fornecedores.

- **Computação**

  É uma das principais funções de um serviço de infraestrutura. Ou seja, processar e computar são as principais capacidades de um computador, como tal, também é fundamental que o fornecedor escolhido provenha de capacidades que atendam às necessidades do cliente. Um fornecedor de *cloud* adequado pode computar imensas tarefas em apenas uns minutos.
  Como podemos observar na análise realizada na tabela 2.3, a Google Cloud não dispõe a possibilidade de implementar aplicações de outras instâncias na sua *cloud*. A Azure Virtual Machines e AWS Lightsail dispõem de um enorme espectro de soluções de computação,

oferecendo uma enorme variedade de sistemas operativos e aplicações. Em suma, são máquinas virtuais com sistemas operativos individuais mas com restrições de acesso aos seus servidores físicos, ou seja operam com servidores virtuais privados. A Google não disponibiliza recursos que realizam a computação em *batches* com alta performance. Por outro lado, a Amazon não possui a possibilidade de gestão de micro-serviços de aplicações. A Google e a Microsoft dispõem de um recurso que facilita o desenvolvimento e a organização de micro-serviços.

Tabela 2.3: Computação

Service	**AWS**	**GCP**	**Microsoft Azure**
Deploy, manage and maintain virtual Servers	*Elastic Compute Cloub (EC2)*	*Compute engine*	*Virtual Machines* *Virtual Machine Scale Sets*
Platform as a Service	*Elastic Beanstalk*	*Google App Engine*	*Cloud Services*
Virtual private servers made easy	*Lightsail*		*Virtual Machines Images*
Ochestrate and manage microservice-bases applications		*App Engine*	*Service Fabric*
Integrate systems and run backend logic processes	*Lambda*	*Cloud Functions*	*Functions* *Event Grid* *Web Jobs*
Run large-scale parallel the high performance batch computing	*Batch*		*Batch*
Automatically scale instances	*Auto Scaling*	*Instance groups*	*Vistual Machine Scale Sets* *Auto Scaling* *App Service Scale Capability*

- **Armazenamento**

  Uma das capacidades mais importantes de serviços de *cloud* é o armazenamento. Com o aumento exponencial da comunicação e consequente crescimento da informação é cada vez mais imprescindível ter um serviço com um poder de armazenamento eficaz e com grande volume.
  Na tabela que se segue, 2.4, podemos observar que a Google não dispõe de recursos de

backup e de *Recovery* ao contrário da Amazon e Microsoft. Assim, sente-se uma integração insegura e volátil de dados no armazenamento da Google.

Tabela 2.4: Armazenamento

Service	**AWS**	**GCP**	**Microsoft Azure**
Object storage service for use cases	*Simple Storage Services(S3)*	*Google Cloud Storage*	*Storage (Block Blob)*
Virtual server disk infrastructure	*Elastic Block Store (EBS)*	*Compute Engine Persistent Disks*	*Storage (Page Blobs)*
Archive storage	*S3 Infrequent Access* *Glacier* *Data archive*	*Nearline* *Coldline*	*Storage (Cool)* *Storage Archive*
Create and configure Shared file systems	*Elastic File System (EFS)*	*ZFS/Avere*	*Files*
Hybrid storage	*Storage Gateway*	*Egnyte Sync*	*StorSimple*
Backup	*Object Storage* *Cold Archive Storage*		*Backup*
Automatic protection and disaster recovery	*Disaster Recovery*		*Site Recovery*

- ***Networking***

  Cada fornecedor oferece diferentes conexões, com características específicas em rede que conectam os seus centros de dados pelo mundo. É notório, na tabela 2.5, que a Google Cloud não provém de recursos para estabelecer uma ligação dedicada e privada. Estas características tornam possível que o utilizador faça uma ligação privada do seu serviço *cloud* para o seu próprio centro de dados ou escritório. É, em muitos casos, uma vantagem pois pode reduzir os custos, aumentar a taxa de transmissão, originar um ambiente mais confiável e fornecer uma ligação mais consistente do que uma ligação via Internet.

Tabela 2.5: Networking

Service	**AWS**	**GCP**	**Microsoft Azure**
Isolated, private cloud, private networking	*Virtual Private Cloud*	*Virtual Private Cloud*	*Virtual Network*
Cross-premises connectivity	*API Gateway*	*Cloud VPN*	*VPN Gateway*
Manage DNS names and records	*Route 53*	*Google Cloud DNS*	*Azure DNS Traffic Manager*
Dedicated, private network connection	*Direct Connect*		*ExpressRoute*

- **Bases de Dados**

Todos os fornecedores oferecem soluções de SQL e NoSQL quando se fala em bases de dados. No contexto desta dissertação, esta característica tem uma importância especial visto que esta inserida no âmbito do *Big Data*. Assim. na tabela 2.6 é facilmente percetível que a AWS e a Microsoft Azure têm características com os mesmos objetivos. Em relação a GCP verifica-se que não existem recursos de migração de bases de dados, este que é importante quando é utilizado mais do que um serviço. Esta característica permite migrar os dados de bases de dados comerciais desde que estas sejam *open-source*, como Oracle,SAP e MySQL, com o menor tempo de inatividade possível. Também não oferece recursos de gestão para o *Data Warehouse*. Recurso este que permite fazer uma análise rápida, eficiente e com custos baixos através de ferramentas de BI e SQL.

Tabela 2.6: Bases de dados

Service	**AWS**	**GCP**	**Microsoft Azure**
Managed relational database-as-a-service	*RDS*	*Cloud SQL* *Cloud Spanner*	*SQL Database* *Database for MySQL* *Database for PostgreSQL*
NoSQL(Indexed)	*DynamoDB*	*Cloud Datastore* *Cloud Bigtable*	*Cosmos DB*
Caching	*ElastiCache*	*Cloud CDN*	*Redis Cache*
Database migration	*Database Migration Service*		*Database Migration Service*
Managed data warehouse	*Redshift*		*SQL Data Warehouse*

- **Segurança**

Neste contexto, os 3 fornecedores providenciam serviços de segurança que são uma das maiores atrações dos utilizadores de *cloud*. Cada um com as suas políticas de segurança que variam conforme a necessidade e o tipo de dados do utilizador. Assim. é visível na tabela 2.7 que a Google se encontra desfalcada no que toca a recursos de segurança. Não fornece *firewall*, que ajuda a proteger de ataques comuns via Internet. Avaliações de segurança que ajuda a melhorar a segurança das aplicações implementadas também não são consideradas no ambiente Google. Da mesma forma, não existe controlo de acessos que ajuda a descobrir identidades maliciosas e permite gerir permissões nas aplicações e não oferece soluções para possíveis ataques.

Tabela 2.7: Segurança

Service	**AWS**	**GCP**	**Microsoft Azure**
Authentication and Authorization	*Identity and Acess Management (IAM)* *Organizations*	*Cloud IAM* *Cloud Identity-Aware Proxy*	*Active Directory* *Active Directory Premium*
Information Protection			*Information Protection*
Protect and safeguard with data encryption	*Key Management Service*		*Storage Service Encryption*
Hardware-based security modules	*CloudHSM*	*Cloud Key Management Service*	*Key Vault*
Firewall	*Web Application Firewall*		*Application Gateway*
Cloud security assessment and certifications services	*Inspector* *Certificate Manager*		*Security Center* *App Service Certificates*
Directory services	*AWS Directory Service*		*Active Directory Domain Services*
Support cloud directories	*Directory Service*		*Windows Server Active Directory*
Cloud services with protection	*Shield*		*DDoS Protection Service*

Capítulo 3

Solução proposta

3.1 Recursos do Azure

Com o objetivo de clarificar o trabalho realizado no âmbito desta dissertação é importante compreender a funcionalidade e a aplicação dos recursos do *Microsoft Azure* utilizados na solução proposta.

Assim, neste capítulo é explicado cada recurso utilizado, tanto na *framework* de monitorização como no caso de estudo realizado.

3.1.1 *Azure Function*

A *Azure function* é um recurso do *Microsoft Azure* que permite correr aplicações simples e sem servidor na *cloud*. É possível implementar uma função que responde às necessidades do projeto sem preocupações com a infraestrutura onde esse código irá correr, como mostrado na figura 3.1. *Azure Function* em geral é desenhada com o objetivo de acelerar e simplificar o desenvolvimento de aplicações.

As funções são uma boa solução para processamento de dados, integração de sistemas, IoT, APIs simples e micro-serviços.

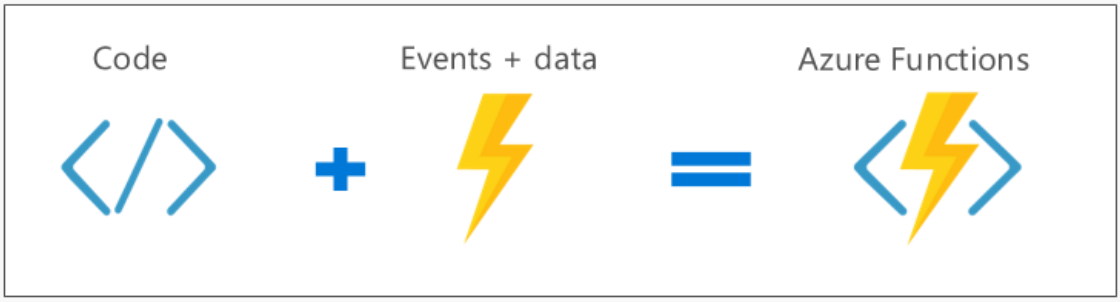

Figura 3.1: Esquema da *Azure Function* [5]

O *Azure Function* provém de várias características que o tornam um recurso tão interessante e diverso. Nas funções é possível escolher a linguagem de desenvolvimento desejada entre C#, F#, Node.js, Java ou PHP, assim o utilizador pode escolher a linguagem que pretende usar consoante a sua vontade e com o que é mais vantajoso para resolver o problema em questão. À semelhança do funcionamento do *Microsoft Azure* este recurso tem a política de pagar por uso, assim só é necessário pagar pelo tempo que o código demora a correr e por cada vez que corre. Esta política torna o uso da *cloud* mais rentável e flexível, pois há oportunidade de escalar os projetos sempre que necessário sem ficar "preso"a uma taxa ou a um preço. Também existem várias bibliotecas referentes às linguagens de programação, como por exemplo NuGet e NPM, que podem ser usadas no *Azure*. Ou seja, mesmo ao programar em *cloud* o utilizador pode usar as bibliotecas que mais lhe convém. Este provém de OAuth em funções *triggered* por pedidos HTTP, que é uma autenticação fornecida por serviços como o *Azure Active Directory*, *Facebook, Google, Twitter* e Conta *Microsoft*. Para implementar funções no *Azure* é possível usar ferramentas de desenvolvimento tais como *GitHub* e *Visual Studio* e mais tarde fazer *deploy* diretamente para o *Azure*.

As funções no *Azure* podem ser *triggered* de várias maneiras de modo a iniciar a execução do código e *bindings*, que são soluções para simplificar o código no que diz respeito à entrada e saída de dados. Ou seja, a função pode ser *triggered* por pedidos HTTP, por tempo, por uma *Blob Storage*, por um *EventHubs*, entre outras.

3.1.2 *Event Hubs*

Event Hubs é um serviço de fluxo de informação e de ingestão de eventos capaz de receber e processar milhões de eventos por segundo. Processa e guarda eventos, dados ou mesmo telemetria produzida por software e aparelhos distintos. Atua como uma fila de espera em tempo real, os eventos são enviados para o *Event Hubs* e este guarda-os e envia-os para outros recursos pela mesma ordem de chegada, garantindo assim a ordem do fluxo de informação. A informação enviada para o *Event Hubs* pode ser analisada em tempo real ou em lotes. Esta é enviada com baixa latência e em grande escala servindo assim como rampa para processamento de *Big Data*.

Um dos objetivos que o *Event Hubs* desempenha em grande parte das soluções que integra é a entrada para um *event ingestor*, como exemplifica a imagem 3.2. Ou seja, este recurso encontra-se entre os produtores e os consumidores de eventos. O *Event Hubs* pode então ter vários produtores de eventos tais como Aplicações, IoT, dispositivos, IP-Capable devices, RTOS, como esquematizado na figura 3.2.

Para enviar eventos para o *Event Hubs* é possível escolher usar AMQP ou HTTPS para cada caso específico. O AMQP requer o estabelecimento de uma socket bidirecional e um nível de segurança de transporte TLS ou SSL/TLS. O protocolo AMQP é mais caro quando se inicializa a sessão no entanto o HTTPS requer SSL adicional por cada pedido. O AMQP tem uma performance mais elevada quando se trata de pedidos frequentes. Quando falamos de consumidores de eventos, o protocolo utilizado é o AMPQ 1.0. Neste caso qualquer entidade que consiga ler eventos do *Event Hubs* é um consumidor de eventos.

O *Event Hubs* garante que todos os eventos que partilham a mesma partição são entregues por ordem e para a mesma partição.

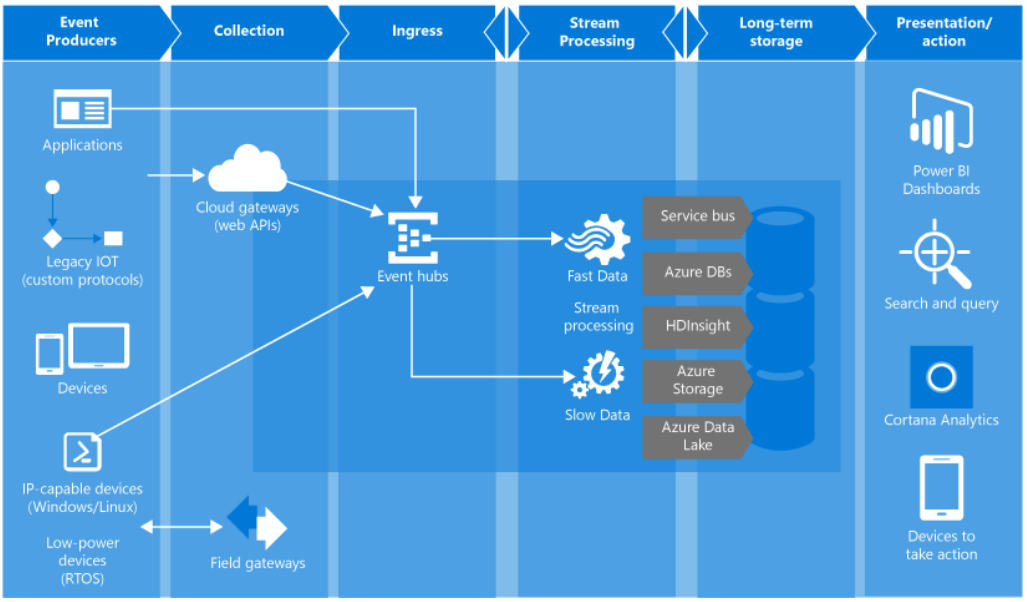

Figura 3.2: Esquema de funcionamento o *Event Hubs* [6]

O *EventHubs* tem certas características que o tornam um recurso diferente de uma fila de espera habitual. Uma característica muito importante no âmbito do problema da dissertação é o *Event Capture*, que permite que os dados da transmissão do *Event Hubs* sejam guardados numa *Blob Storage*. Esta característica é importante pois permite a monitorização dos eventos que passam no *Event Hubs*. Uma outra característica que o distingue de filas de espera tradicionais são as partições que permitem que cada consumidor leia apenas um subconjunto de informação específico. O *Event Hubs* garante assim que todos os eventos que partilham a mesma partição são entregues por ordem e para a mesma partição. Uma partição tem capacidade de uma *throughput unit*, unidade que quantifica a capacidade do EventHubs:

- Entrada: Até 1 MB por segundo ou 1000 eventos por segundo
- Saída: Até 2 MB por segundo

Também os produtores de eventos provêm de *Tokens SAS* que identifica e autentica o editor do evento. Com o objetivo de usar a informação que passa no *Event Hubs* para vários grupos, existem grupos de consumidores que oferece uma vista distinta do fluxo de dados permitindo que os consumidores sejam independentes.

3.1.3 *Stream Analytics*

O *Stream Analytics* (SA) é um recurso essencial na construção da *framework* proposta no âmbito da dissertação. É um recurso que permite a análise de grandes volumes de dados em

tempo real, vindos de dispositivos, sensores, *web sites*, redes sociais, aplicação, entre outros. Para fazer análise de dados no SA, o utilizador recorre a vários tipos de ferramentas disponibilizadas pelo mesmo.

Com o intuito de examinar a informação que chega ao *stream analytics* é necessário criar um *stream analytics job* onde é definido quais as entradas e saídas do *job*. *Azure Stream Analytics job* tem como possíveis entradas de *stream* o *Azure Event Hub*, *Azure IoT Hub* e a *Blob Storage*. Entradas estas que têm produtores de eventos tais como dispositivos e aplicações variadas. Como entrada de referência existe a *Blob Storage* que representa a informação estática ou quase estática do sistema. Para usar a informação proveniente da entrada de referência é necessário cruzá-la com uma entrada de *stream*, usando o comando T-SQL, JOIN. O *job* específica também a *query* que transforma a informação vinda das entradas do *job* de acordo com o que queremos visualizar na saída. Esta *query* de transformação usa um *subset* de T-SQL específico do SA que é usada para filtrar, ordenar, agregar e juntar data em tempo real sobre um período de tempo. Depois de analisada, a informação vai ser enviada para os outputs do *Stream Analytics* onde vai ser visualizada e/ou guardada, na figura 3.3 temos os outputs possíveis do *stream analytics*, tais como *Azure Function*, *Azure Cosmos DB*, *Power BI*.

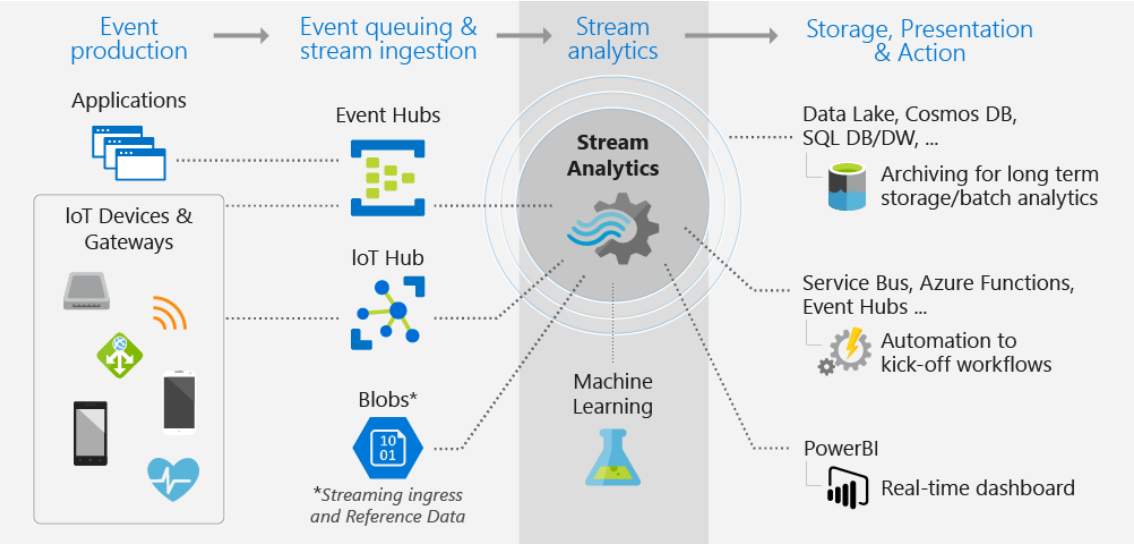

Figura 3.3: Esquema de funcionamento do *Stream analytics* [7]

O *Stream analytics* é um recurso com muitos benefícios, é fácil de usar inicialmente, é simples de definir entradas e saídas e é percetível a maneira como as funcionalidades estão ligadas. Com o *Stream analytics* é possível escalar o projeto a qualquer momento. Os custos são relativamente baixos, pois é pago por uso facilitando assim mudanças nos projetos e aumentando a performance do recurso.

3.1.4 *Blob Storage*

Uma *storage account* é uma solução moderna de armazenamento de dados. O *Azure storage account* é concebido para ser altamente escalável para conseguir responder às necessidades dos dias de hoje. A redundância dos dados garante que estes estão seguros em caso de falhas de hardware transitórias. É possível assim, com o mesmo intuito, a replicação de dados em *datacenters* de regiões geográficas diferentes evitando que haja perda de dados em situações de desastre. Assim, os dados estão sempre disponíveis para o uso do utilizador. Sendo a segurança um dos requisitos mais importantes quando se fala em armazenamento de dados, o *Azure* garante que toda a informação escrita no *Azure Blob Storage* é encriptada pelo próprio serviço. Adicionalmente oferece ao utilizador o total controlo sobre quem acede aos seus dados.

Dentro da *Microsoft Azure Storage Account* existem 4 serviços de armazenamento de dados diferentes. Como podemos verificar na figura 3.4, existe *Azure Files*, *Azure Queues*, *Azure Tables* e *Azure Blobs* que é o serviço mais utilizado no âmbito desta dissertação.

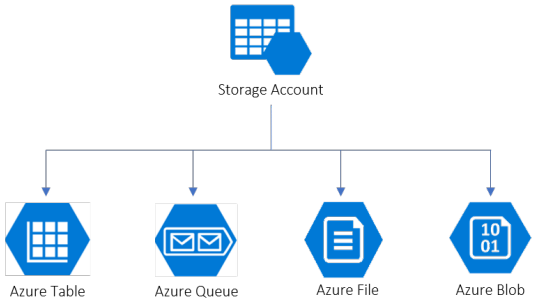

Figura 3.4: Esquema da *Storage account* (Elaboração própria)

A *Azure Blob Storage* tem como objetivo o armazenamento de todos os tipos de dados. Guarda grandes quantidades de informação estruturada e não estruturada, seja ela texto ou binária, o que a torna muito versátil a qualquer projeto. Também da mesma forma é compatível com muitos recursos do *Azure*, incluindo o *Stream analytics* que é o foco central desta dissertação. Esta é acessível em qualquer sitio via HTTP ou HTTPS. A *Blob Storage* está armazenada em *containers*, como é mostrado na figura 3.5, definidos pelo utilizador e estes *Containers* estão dentro de uma *Storage Account* criada inicialmente pelo utilizador.

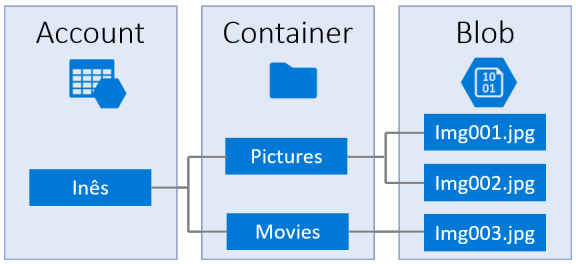

Figura 3.5: Esquema da *Blob Storage* (Elaboração própria)

3.1.5 *Data Factory*

É um serviço de integração de dados baseado em *cloud* que permite criar fluxos de trabalho orientados a dados na *cloud* que orquestram e automatizam a movimentação e transformação de dados, ou seja é uma espécie de processo de ETL em *cloud*. É um serviço que permite retirar informação tanto de repositórios da *cloud* como de repositórios locais, podendo se então denominar como um serviço híbrido. Característica esta essencial pois assim é possível fazer a extração na *cloud* de várias fontes externas e internas.

Por exemplo no caso de estudo de alerta para fraudes, descrito no capítulo 1.3, para saber as informações pessoais do cliente como o número de telefone que é necessário no problema em questão, é necessário consultar o DW local da empresa usando o *Data Factory*. Por outro lado, as transações que chegam ao *Azure* estão em *cloud* e vão ser cruzadas com a informação proveniente do DW, com o intuito de oferecer uma melhor experiência aos consumidores.

Resumindo, é possível extrair informação de servidores locais ou da *cloud*, referentes ao processo de extração. Durante o processo é possível fazer transformações à informação de acordo com as necessidades do projeto, processo de transformação. Por último, esta informação é carregada para a *cloud*, como por exemplo para uma *Blob Storage*, *SQL Database*, entre outras. Este último processo corresponde ao carregamento. É possível visualizar o seguinte processo na figura 3.6.

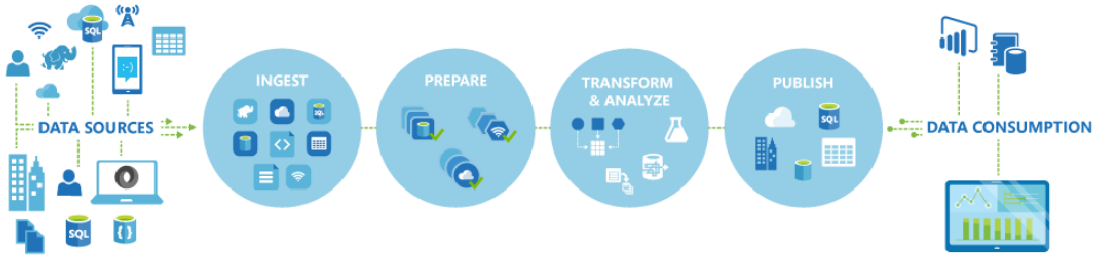

Figura 3.6: Esquema do processo do *Data Factory* [8]

Então com o uso do *Data factory* é possível:

- Criar e calendarizar *workflows*, conhecidos como *pipelines*, que recebem dados de várias fontes diferentes;
- Processar e transformar a informação usando serviços de computação como *Azure HDInsight Hadoop, Spark, Azure Data Lake Analytics,* e *Azure Machine Learning*;
- Publicar dados de saída em servidores de cloud, como o *SQL Data Warehouse do Azure* para aplicações de *Business intelligence* (BI);

3.1.6 *Preços dos recursos*

Neste subcapítulo vão ser analisados os preços dos recursos do *Microsoft Azure* usados no âmbito da dissertação descrita, com o objetivo de fazer um orçamento final da *framework* de monitorização a ser realizada.

- ***Event Hubs***

  Na tabela 3.1 podemos verificar os preços aplicáveis ao uso do recurso *Event Hubs*.

Tabela 3.1: Preços do *Event Hubs*

	Básico	**Standard**	**Dedicado**
Eventos de entrada	€0,024 por milhão de eventos	€0,024 por milhão de eventos	Incluído
Unidade de débito (entrada de 1 MB/s, saída de 2 MB/s)	€0,013/hora	€0,026/hora	Incluído
Tamanho da mensagem	256 KB	256 KB	1 MB
Políticas do publicador	Não disponível	Disponível	Disponível
Grupos de consumidores	1 - Predefinição	20	20
Repetição de mensagens	Disponível	Disponível	Disponível
Unidades de débito máximas	20	20	1 CU aproximadamente 50
Retenção de mensagens	1 dia incluído	1 dia incluído	Até 7 dias incluídos
Captura	Não disponível	€0,085/hora	Incluído

- ***Stream Analytics***

  Este recurso só cobra o utilizador pelo tempo em que existem eventos de entrada, como podemos verificar na tabela 3.2.

Tabela 3.2: Preços do *Stream Analytics*

Uso	**Preço**
Eventos de entrada	€0,102/hora

- ***Blob Storage***

  A redundância de dados é uma das características mais interessantes que o *Microsoft Azure* oferece. Esta consiste em ter um sistema com alta disponibilidade o que garante que os dados estejam disponíveis em qualquer situação. Com esse intuito os dados são taxados conforme o tipo de replicação pretendida por parte do cliente, como podemos ver na tabela 3.4.

Tabela 3.3: Redundância de dados

Armazenamento localmente redundante (LRS)	Concebido para garantir pelo menos, 99.99999999999% (11 noves) de durabilidade dos objetos ao longo de um determinado ano ao manter várias cópias dos seus dados num único *datacenter*
Armazenamento com redundância de zona (ZRS)	Concebido para garantir, pelo menos, 99,999999999999 %(12 noves) de durabilidade dos objetos ao longo de um determinado ano ao manter várias cópias dos seus dados em vários *datacenters* ou regiões.
Armazenamento geograficamente georredundante (GRS)	Concebido para garantir, pelo menos, 99,9999999999999999 %(16 noves) de durabilidade dos objectos ao longo de um determinado ano ao manter várias cópias dos dados numa única região, replicando de forma assíncrona para uma segunda região.
Armazenamento geograficamente georredundante com acesso de leitura (RA-GRS)	Concebido para garantir, pelo menos, 99.9999999999999999 % (16 noves) de durabilidade dos objetos ao longo de um determinado ano e 99,99 % de disponibilidade de leitura ao permitir acesso de leitura a partir da segunda região utilizada para o GRS.

Assim, no projeto desenvolvido no âmbito da dissertação em questão é usado armazenamento localmente redundante. Para este plano de redundância temos os preços descritos nas tabelas

3.4, 3.5 referentes à *Blob Storage*. Na tabela seguinte, é visto o preço de armazenamento de dados. Os preços variam de acordo com o tipo de *blob*: Acesso frequente, Acesso esporádico e Arquivo. É necessário também ter em conta que os preços descritos na tabela 3.4 são por GB, por mês.

Tabela 3.4: Preços relativos ao armazenamento de dados na *Blob Storage*

	Acesso frequente	**Acesso esporádico**	**Arquivo**
Primeiros terabyte (TB) 50/ mês	€0,01666 por GB	€0,0085 por GB	€0,0019 por GB
Próximos 450 TB/Mês	€0,0159 por GB	€0,0085 por GB	€0,0019 por GB
Mais de 500 TB/Mês	€0,0153 por GB	€0,0085 por GB	€0,0019 por GB

Na tabela 3.5 é possível observar os preços referentes às operações e transferência de dados na *Blob Storage*.

Tabela 3.5: Preços referentes às operações e transferência de dados na *Blob Storage*

	Acesso frequente	**Acesso esporádico**	**Arquivo**
Operações de Escrita (por 10 mil)	€0,05	€0,08	€0,10
Operações de Listagem e de Criação de Contentores (por 10 mil)	€0,000	€0,05	€0,05
Operações de Leitura (por 10 mil)	€0,00	€0,01	€5,06
Todas as outras Operações (por 10 mil), exceto a Eliminação, que é gratuita	€0,000	€0,00	€0,00
Obtenção de Dados (por GB)	Gratuito	€0,01	€0,02
Escrita de Dados (por GB)	Gratuito	Gratuito	Gratuito

- ***Azure Function***

A *Azure Function* cobra com base no número total de execuções de todas as funções por mês. As execuções são contadas cada vez que uma função é executada em resposta a um evento. O primeiro milhão de execuções de cada mês é gratuito.

Estas são também cobradas com base no consumo de recurso observado, este que é medido em GB por segundos. O consumo de recursos observado é calculado pela multiplicação do tamanho médio da memória em GB pelo tempo em milissegundos necessário para executar a função. A

memória usada por uma função é medida arredondando para o 128 MB mais próximo até o tamanho da memória máximo de 1.536 MB, com o tempo de execução calculado arredondando para o 1 ms mais próximo. O mínimo de memória e de tempo de execução de uma única execução de função é de 128 MB e 100 ms, respetivamente. O preço da função inclui uma concessão gratuita de 400.000 GBs. É possével verificar os preços na tabela 3.6.

Tabela 3.6: Preços referentes à *Azure Function*

Medidor	**Preço**	**Concessão Gratuita (Por Mês)**
Tempo de Execução	€0,000014/GB/s	400.000 GB/s
Total de Execuções	€0,169 por milhões de execuções	1 milhão de execuções

3.2 Casos de estudo sobre alerta de fraudes

Para responder à necessidade exposta no capitulo 1.3, foi realizado um projeto no *Microsoft Azure*. A solução foi desenhada pela empresa em questão e implementada pela B2F. Esta solução usa vários recursos, como *Azure Function*, *EventHubs*, *Stream analytics*, *Blob Storage*, *Table Storage* e *Data Factory*.

A *Azure Function* está responsável por receber informação do cliente que vem em formato XML pelo protocolo SOAP. Como o *Event Hubs* não recebe XML com a estrutura do protocolo SOAP foi implementada uma função que recebe essa informação, a transforma em formato JSON e retira os cabeçalhos referentes ao protocolo. De seguida, essa informação é enviada para um *EventHubs* que é o intermediário entra a *Azure Function* e o *Stream analytics*, criando uma fila de espera.

Com o objetivo de cruzar informação, neste processo específico, é necessário recolher informação de um *Data Warehouse*, onde está presente toda a informação dos clientes, necessária para a realização do processo, como por exemplo o número de telefone que é imprescindível pois o objetivo final do processo é que o cliente receba uma SMS a alertar aquando de uma transação fraudulenta. Então usando um *Data Factory* é possível aceder ao DW da instituição financeira e copiar os dados necessários para uma *Blob Storage* diariamente.

O *Stream analytics* vai ter duas entrada, uma *stream* e outra de referência. Na entrada em *stream* os dados da transação provenientes da API e na entrada de referência os dados do DW referentes à informação pessoal do cliente. Estes são cruzados através do número de conta sendo assim possível associar a transação a um cliente específico.

Com o objetivo de se fazer uma análise estatística o *Stream Analytics* tem como saída o *Power BI* onde são observadas estatísticas das transações em tempo real. Da mesma forma, para existir um histórico dos dados recebidos da API externa, o SA tem uma saída para uma *Table storage* onde as transações são guardadas e podem ser consultadas manualmente sempre que for necessário. Para se concretizar o objetivo final do processo foi criada uma saída no *Stream analytics* para uma

função que envia por um pedido HTTP a informação necessária para o envio da mensagem ao cliente por meio de uma API externa. É possível visualizar o esquema no sistema de deteção de fraudes na figura 3.7.

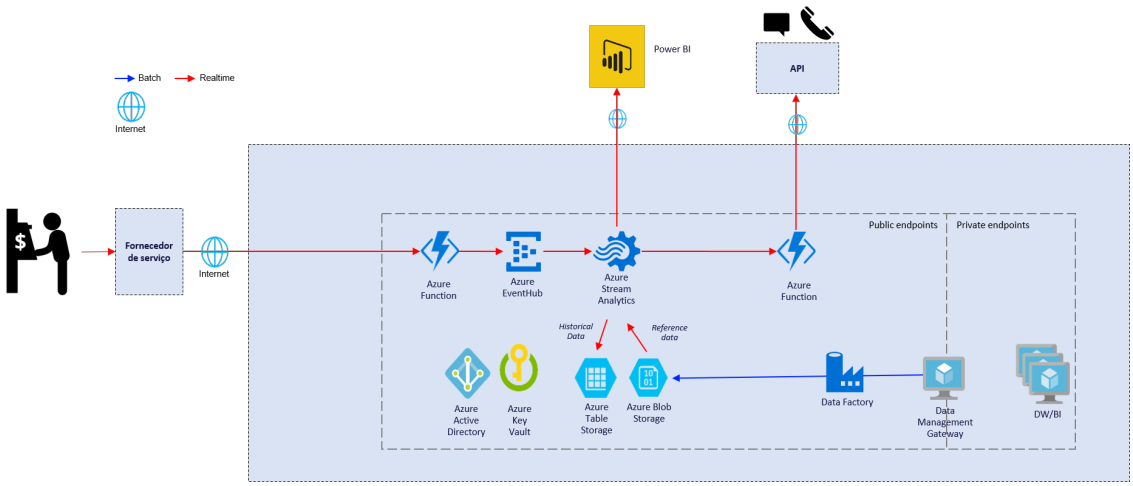

Figura 3.7: Esquema do projeto de alerta de fraudes (B2F

3.3 *Framework* de monitorização

A *framework* de monitorização foi idealizada separadamente, sendo dividida em 4 pontos. Primeiramente, foi realizado um processo que simula o caso de estudo de alerta de fraudes. De seguida, foi construida a monitorização referente à qualidade dos dados seguida da monitorização de respostas da API e finalmente à monitorização referente aos *logs* de diagnóstico.

3.3.1 Simulação do caso de estudo de alerta de fraudes

Com base no projeto de alerta de fraudes foi realizado um processo para integrar a *framework* de monitorização. Esse projeto consiste em receber a transação, ou seja a transação vai ser recebida através de um protocolo HTTP por uma *Azure function*. Depois disso, vai ser enviada para um *Event Hub* e seguidamente para o *Stream Analytics*. No *Stream Analytics* ela vai ter uma saída para uma *blob* e outra para o *Power BI*. A saída referente à *blob* é utilizada como prova de conceito pois poderia ser outra saída qualquer desde que compatível com o SA. Assim, é possível ver os dados que chegam ao *Stream analytics* no *Power BI* e guardá-los numa *blob*. O esquema encontra-se na figura 3.8.

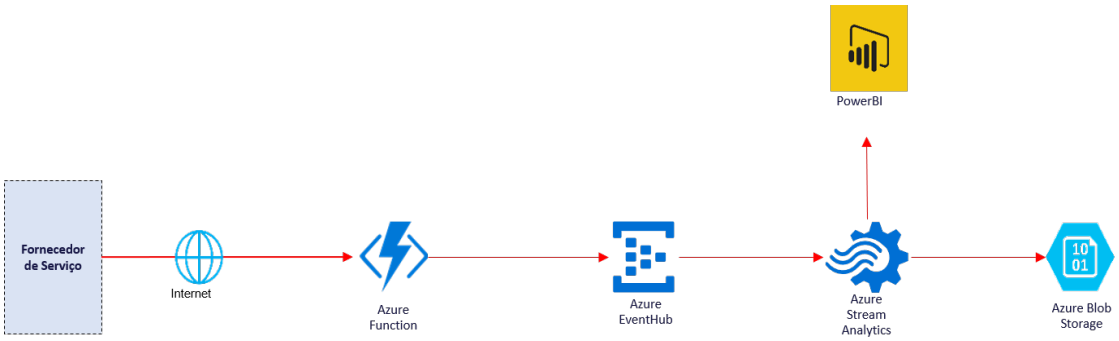

Figura 3.8: Esquema da simulação do caso de estudo de alerta de fraudes (Elaboração própria)

Este processo foi realizado como prova de conceito para a *framework* de monitorização, pois trata-se de um caso simples e geral que pode ser flexível a outros problemas.

3.3.2 Qualidade dos Dados

Durante a realização do caso de estudo de alerta de fraudes foram encontrados alguns possíveis erros no processo, que necessitam de uma monitorização automática sem necessidade de adquirir uma pessoa que monitorize manualmente. A *framework* referente à qualidade de dados é focada na possível perda de informação entre recursos. Ou seja, se alguma transação se perder ao longo do percurso a *framework* deve ser capaz de detetar essa falha, a hora, o campo em falta e o messageID correspondente. Assim, com esse intuito foram enumeradas as possíveis perdas de informação no processo, também explicado na figura 3.9:

1. Receber informações incompletas do lado do cliente;

2. Perder informação no envio de transações da *Azure Function* para o *EventHubs*;

3. Perder informação no envio de transações do *Event Hubs* para o *Stream Analytics*;

4. Perder informação no envio de transações do *Stream Analytics* para a *Blob Storage*;

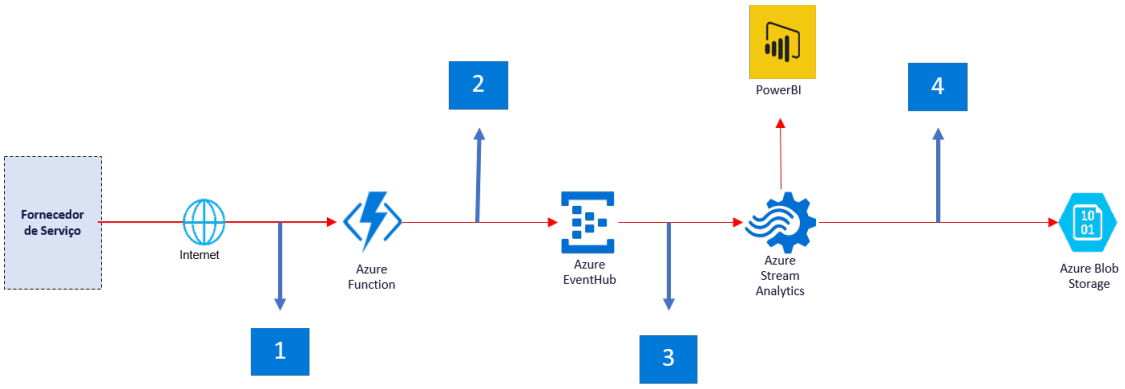

Figura 3.9: Esquema com os pontos de possíveis perdas de dados (Elaboração própria)

Assim, verifica-se que existem 4 pontos do processo onde é possível que o fluxo de informação se adultere, como mostrado na figura 3.9.

Com o objetivo de colmatar estas possíveis falhas foi pensada uma *framework* de monitorização de qualidade de dados. Esta *framework* consiste em guardar o conteúdo de cada recurso num *container* de uma *Blob Storage*, como mostrado na figura 3.10, para ser possível obter o histórico das transações que atravessam cada recurso. Essas *blobs* são atualizadas cada vez que passa uma transação num recurso, ou seja em tempo real.

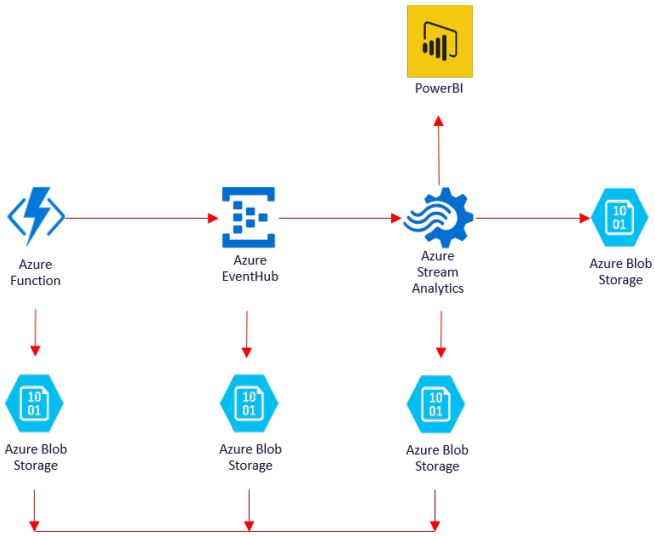

Figura 3.10: Esquema da monitorização da Qualidade de dados (Elaboração própria)

De seguida estas *blobs* vão ser definidas como entradas de *stream* no *stream analytics*, 3.11. Desta forma é possível monitorizar em tempo real as transações. Essa monitorização vai ser visualizada no *Power BI* onde são gerados gráficos e métricas.

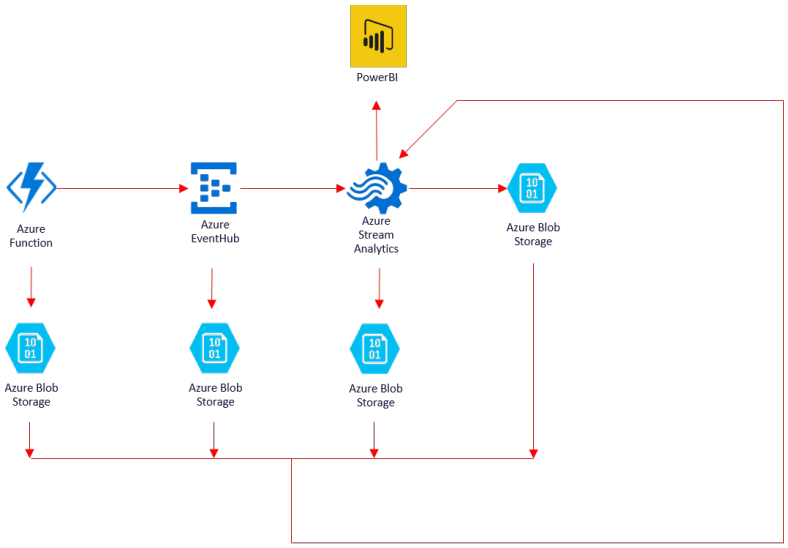

Figura 3.11: Esquema da monitorização da Qualidade de dados completo (Elaboração própria)

3.3.3 Resposta da API

Visto que no processo de alerta de fraudes é usada uma API externa é importante monitorizar a resposta da API aquando a chamada da mesma. É necessário saber se esta foi chamada com sucesso ou se ocorreu um erro. Se ocorrer erro é importante perceber o tipo de erro para atuar da maneira mais rápida e eficaz possível. Para isso foi criado então uma lista com as possíveis respostas HTTP, que se encontra no anexo B. Os códigos referentes ao *Status Code* da resposta da API são separados em 5 grupos de acordo com o primeiro número do código.

- 1xx — Informacional.
- 2xx — O pedido foi realizado com sucesso.
- 3xx — O cliente é redirecionado para um recurso diferente.
- 4xx — O pedido contém um erro de algum tipo.
- 5xx — O servidor encontrou um erro ao realizar a consulta.

Então, com o intuito de monitorizar essa resposta foram criadas duas *Azure Functions*. Uma que simula o comportamento normal de uma API, ou seja quando recebe a informação vinda do pedido responde com o *Status Code OK* e quando não recebe responde com um *Status Code BadRequest*. Este *Status Code* pode ser mudado para efeitos de teste, para qualquer tipo de resposta por parte da API. A outra função chama a API, onde manda informação teste, e lê a resposta da mesma. Se a resposta por parte da API não for de sucesso esta função tenta mandar a informação teste até 3 vezes até existir um retorno de sucesso. Esta função vai também guardar a resposta numa *Blob* e envia-a para o *Stream Analytics* onde vai ser analisada e posteriormente visualizada no *Power BI*.

3.3.4 Diagnóstico de Logs

Com o objetivo de perceber o comportamento dos recursos ao longo do processo foi feita uma monitorização dos *Logs* dos recursos utilizados no projeto. Por essa razão, foi estudado os tipos de *Logs* possíveis no *Microsoft Azure*. Estes podem ser:

- ***Control/management logs***.

  Um exemplo disso são os *Azure Activity Logs*.

- ***Date plane logs***.

  Por exemplo os *Azure Diagnostic Logs*

- ***Processed events***.

  Como é exemplo o Azure Security Alerts

Os *Activity Logs* fornecem informações sobre as operações que foram executadas nos recursos da subscrição, por exemplo criar um *Stream Analytics* ou apagar um *Event Hubs*. Ou seja são *logs* ao nível da subscrição, dando informações sobre os eventos que ocorrem numa subscrição específica.

Os *Diagnostic Logs* são *Logs* ao nível do recurso. Ou seja, fornecem conhecimento sobre as operações realizadas no recurso específico, como por exemplo correr o *job* do *Stream Analytics*.

O *Azure Security Alerts* extrai, analisa e integra automaticamente dados de *logs* dos recursos do *Azure*, da rede ou de soluções, como *firewall* e soluções de proteção de *endpoint*, para detetar ameaças reais.

Feita a análise dos possíveis logs do *Microsoft Azure*, foi escolhido monitorizar os *Diagnostic Logs* visto que são os mais adequados ao problema. Estes fazem uma monitorização ao nível dos recursos que cumpre o objetivo pretendido no contexto desta dissertação. Que consiste em perceber o que acontece internamente com os recursos. Existem registo de *diagnostic Logs* em vários recursos do *Azure*, no processo descrito ao longo deste documento só existem *logs* para o *Azure Stream Analytics* e para o *Event Hubs*. Na tabela 3.7 é possível ver os recursos que provêm de *diagnostic logs* e o respetivo serviço, esquema e categoria.

Tabela 3.7: Recursos que têm *Diagnostic Logs*

Serviço	Esquema	Tipo de Recurso	Categoria
Event Hubs	Azure Event Hubs diagnostic logs	Microsoft.EventHub/namespaces	ArchiveLogs
		Microsoft.EventHub/namespaces	OperationalLogs
Stream analytics	Job diagnostic logs	Microsoft.StreamAnalytics/streamingjobs	Execution
		Microsoft.StreamAnalytics/streamingjobs	Authoring

Do lado do *Event Hubs* existem dois tipos de *logs* possíveis:

- **Archive Logs**

  São registos relacionados com o *Event Hubs* especificamente com os registos relacionados com erros de arquivo.

- **Operational Logs**

  Informações sobre o que acontece durante as operações do *Event Hub* mais especificamente operações do tipo criação do *Event Hub*, os recursos utilizados e o estado da operação.

Por outro lado o *Stream analytics* pode também ter dois tipos de *logs* diferentes:

- **Authoring Logs**

  Diz respeito a registos relacionados com autorizações de operações do *Job*, tais como a criação de um *Job*, adicionar entradas ou saídas ao *Job*, fazer alterações na *query*, começar ou parar o *Job*.

- **Execution Logs**

  Regista eventos que ocorrem durante a execução do *Job*. Esses eventos podem ser erros de conectividade, erros de processamento dos dados, entre outros.

Os *Diagnostic Logs* provêm de soluções para exportar os dados com o objetivo de os monitorizar, como podemos ver na figura 3 existe a opção de exportar os dados para uma *blob*, para um *Event Hubs* ou para um *Archive Storage*. No contexto desta dissertação foi decidido que os dados iam ser exportados para um *Event Hubs*, para serem analisados no *Stream Analytics* e mais tarde visualizados no *Power BI*.

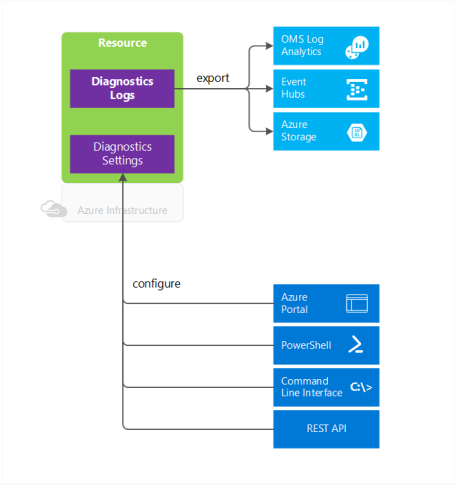

Figura 3.12: Esquema de soluções possíveis para extrair os *Diagnostic logs* [9]

3.3.5 Resultado final

Neste subcapítulo é possível observar o esquema completo da *framework* de monitorização implementada no âmbito da dissertação descrita neste documento, figura 3.13.

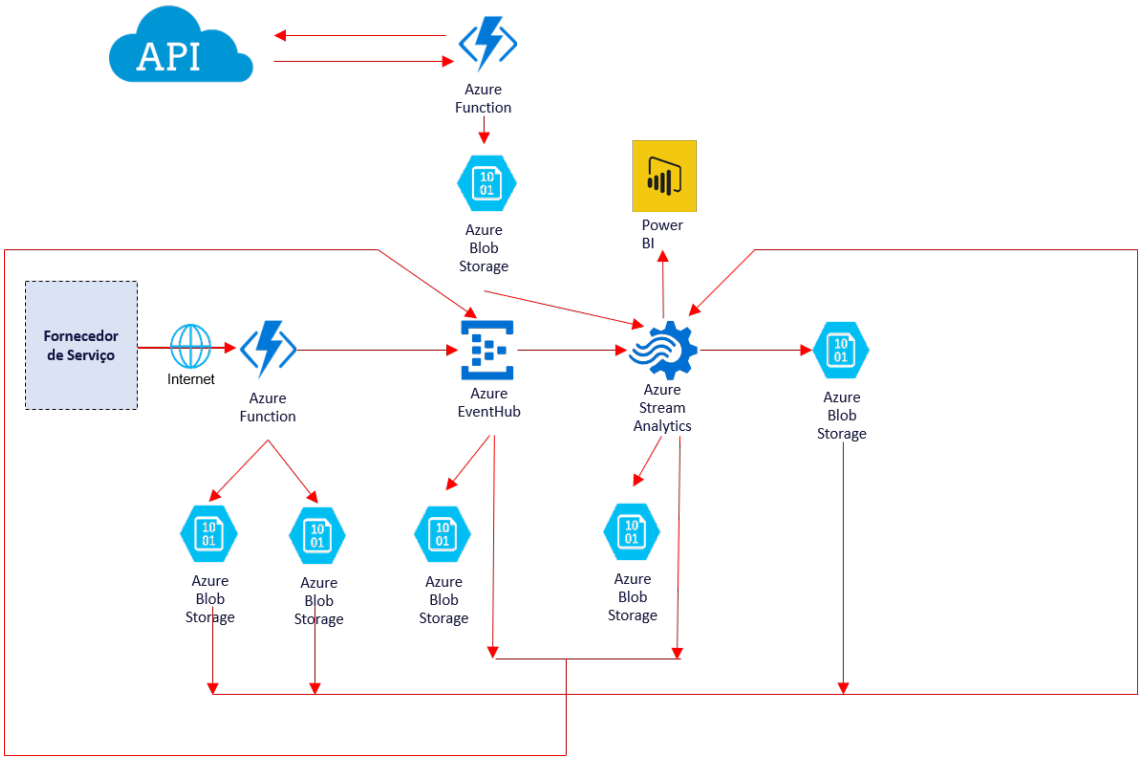

Figura 3.13: Esquema da *framework* de monitorização (Elaboração própria)

Capítulo 4

Implementação

Para a implementação da *framework* de monitorização apresentada no capítulo 3 foram utilizadas técnicas de *Cloud computing*, usando o Microsoft Azure. É uma plataforma que pode ser usada em qualquer computador desde que tenham ligação com *Internet*. No presente capítulo é explicado como é que a implementação do *framework de monitorização* foi realizada incluindo igualmente o caso de estudo realizado. Para facilidade de implementação a *framework* foi implementada por partes começando por desenvolver a monitorização de qualidade dos dados de seguida a resposta da API e por último os *logs* de diagnósticos.

4.1 *Framework* de monitorização

4.1.1 Simulação do caso de estudo de alerta de fraudes

Para a implementação da simulação do caso de estudo de alerta de fraudes foram utilizados os seguintes recursos do *Microsoft Azure*: *Azure Function*, *Event Hubs*, *Stream Analytics*, *Blob Storage*. Na figura 4.1 esta representado o esquema da simulação do caso de estudo de alerta de fraudes que é descrito de seguida.

A *Azure Function* recebe por meio de um pedido HTTP uma transação em XML num protocolo SOAP, pode ser vista a sua estrutura no anexo A. Na *Azure function* a transação é transformada do formato XML para o formato JSON e posteriormente são retirados os *headers* específicos do protocolo SOAP. Posteriormente, a informação é guardada num dicionário, onde foi construida uma rotina que retira os cabeçalhos Key/Value da transação. Depois de tratada, a transação é enviada para o *Event Hubs*, em formato JSON, usando uma *connection string* definida ao criar o *Event Hubs*.

Seguidamente, foi definido no *Stream analytics* uma entrada referente ao *Event Hubs*, que é responsável por receber os dados proveniente do mesmo no *Stream analytics*. As transações recebidas no SA são enviadas para uma *Blob Storage* e para o *Power BI*. Para isso foi necessário definir saídas no *Stream analytics* referentes a uma *blob storage* a ao *Power BI*. Assim, as transação são visualizadas no *Power BI* bem como métricas sobre as mesmas.

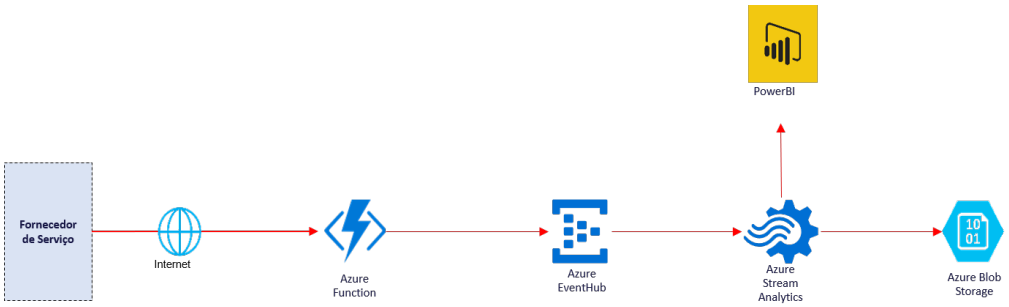

Figura 4.1: Esquema da simulação do caso de estudo de alerta de fraudes (Elaboração própria)

4.1.2 Qualidade dos dados

Para a implementação da monitorização da qualidade dos dados foram usados recursos como *Azure Function, Azure EventHub, Azure Stream Analytics, Azure Blob Storage* e o *PowerBI*. Cada um deste recursos são parte integrante da *framework* de monitorização e apresentam funções e aplicações diferentes.

Primeiramente, foi criada uma rotina na *Azure function*, adicionando à função já implementada anteriormente, que verifica cada campo de uma transação e verifica se o campo recebido se encontra preenchido ou não. Então, se a transação se encontrar com campos em falta a função vai guarda-la numa pasta da *Blob Storage* específica. No caso contrário, a transação vai ser enviada para o *Azure EventHubs* e guardada numa pasta distinta da *blob*. Assim, é evitado que informação corrompida continue no processo diminuindo possíveis erros na continuidade do mesmo.

Concluindo, vai existir uma pasta na *blob storage* com transações defeituosas e uma outra com transações completas, como representado na figura 4.2.

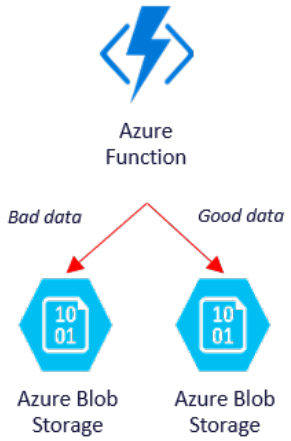

Figura 4.2: Esquema da *framework* para a qualidade de dados-Parte1 (Elaboração própria)

De seguida, os ficheiros que não se encontram danificados vão da *Azure Function* para um *Event Hubs*, figura 4.3. Para isso foi necessário configurar a função para mandar um pedido HTTP para o *Event Hubs*, usando uma *connection string* fornecida por uma *shared acess policy*, criada na altura de criação do *Event Hubs*. O *Event Hubs* tem uma funcionalidade, chamada *capture*, que permite guardar numa pasta da *blob* todas as mensagens que chegam ao *Event Hubs*, podendo assim monitorizar se as transações passam esta fase com sucesso. Por razões orçamentais não foi possível implementar esta funcionalidade. Contudo, foi realizado um teste onde se verifica que é possível realizar este tipo de processo.

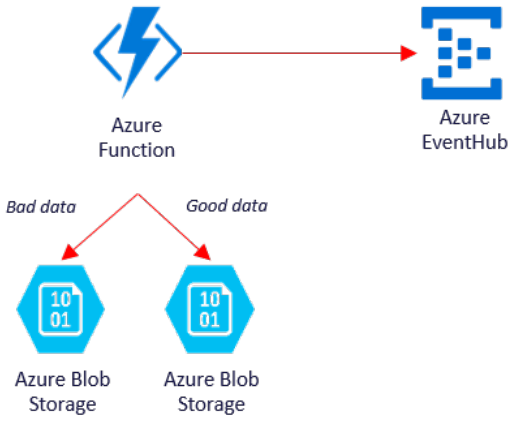

Figura 4.3: Esquema da *framework* para a qualidade de dados-Parte2 (Elaboração própria)

Depois da integração com o *Azure Event Hubs*, vem o *Azure Stream Analytics*. Este que vai receber, analisar e reencaminhar as transações provenientes da parte posterior do processo em tempo real. Ou seja, neste caso o *Stream Analytics* vai ter como entrada o *Event Hubs* e as pastas da *blob* de monitorização e como saída o *Power BI* e uma *blob*, onde a *blob* final representa apenas o ponto final do processo e o *Power BI* é onde os dados da monitorização são visualizados. Para isso, definimos então como entradas de *stream* o *EventHubs*, uma pasta da *blob* com as transações iniciais completas, outra com as transações iniciais com defeitos e uma que representa as transações que chegam ao fim do processo. Na figura 4.4 podemos verificar o design da solução completa como descrito anteriormente.

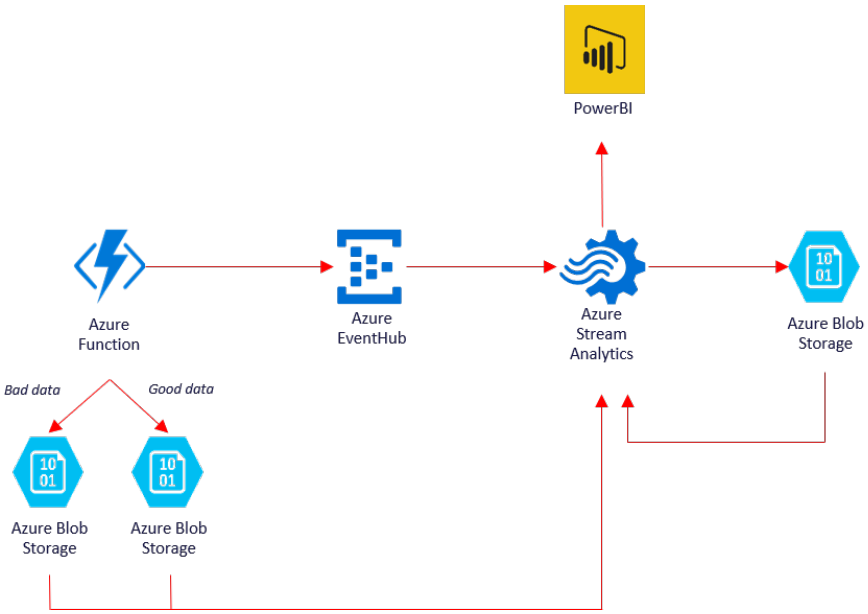

Figura 4.4: Esquema da monitorização da qualidade dos dados (Elaboração própria)

Com o objetivo de fazer uma análise no *Stream analytics* de valor para uma *framework* de monitorização foram comparadas as entradas do *Stream analytics*. Estas comparações são feitas numa *query* do *Stream analytics* que provém de uma linguagem própria chamada *Stream Analytics Query Language*, que é um sub-grupo de T-SQL. Por exemplo quando é necessário fazer um JOIN entre duas transações é necessário definir um DATEDIFF, que é o intervalo de tempo em que duas transações podem ser desfasadas uma da outra. Estas diferenças são essencialmente devidas à *query* ser realizada em tempo-real, onde está sempre a receber transações. Então a tabela 4.1 representa um pequeno resumo da query do *stream analytics*.

Tabela 4.1: tabela representativa da *Query* do *stream analytics* da parte da monitorização da qualidade dos dados

Fonte de Informação 1	**Fonte de Informação 2**	**Objectivo**
Blob Storage inicial (Com informação completa)	Blob Final	-Verificar quantos transações existiam inicialmente na Blob inicial; -Verificar quantas transações chegaram à Blob Final; -Contar o número de insucessos;
Blob Storage inicial (Com informação incompleta)		-Verificar quantas transações foram danificadas; -Verificar em cada transação quantos campos não estão preenchidos; -Verificar que campos foram mais vezes corrompidos; -Verificar a que horas existiram mais transações danificadas;
Blob Storage inicial (Com informação completa)	Informação que chega ao Stream analytics	-Verificar se não se perde informação durante o EventHubs; -Verificar se há transações que não chegam ao Stream analytics; -Fazer uma contagem de sucessos e Insucessos;
Blob Storage inicial (Com informação incompleta)	Informação que chega ao Stream analytics	-Verificar se informação corrompida não passa para o resto do processo;

Na eventual expansão deste projeto, foi feito um caso de estudo onde se considerou a monitorização de todos os recursos possíveis que envolvem o *Stream analytics*. Desta forma, é possibilitada a escalabilidade desta *framework* independentemente das entradas e saídas que o *stream analytics*

possa ter. Usando a mesma metodologia pensada anteriormente toda a informação que passa nos recursos é guardada numa *blob storage* e enviada novamente para o *stream analytics*, como esquematizado na figura 4.5. Ao longo da implementação desta simulação apareceram alguns obstáculos em relação à mesma. Existem alguns recursos onde não é possível enviar a informação imediatamente para uma *blob*, como o *Data Lake Store*, a *SQL Database* e a *Azure Table Storage*. Estes precisam de um recurso intermédio, denominado *Data factory*, que tem um tempo mínimo de transmissão de 15 minutos e tem um custo elevado. Ou seja, os dados não iam ser controlados em tempo real o que ia adulterar as tomadas de decisões pensadas anteriormente e iria aumentar razoavelmente o preço da solução, o que não é desejável aquando a realização de uma *framework* de monitorização. Esta *framework* foi então uma prova de conceito, para uma *framework* de uma dimensão maior.

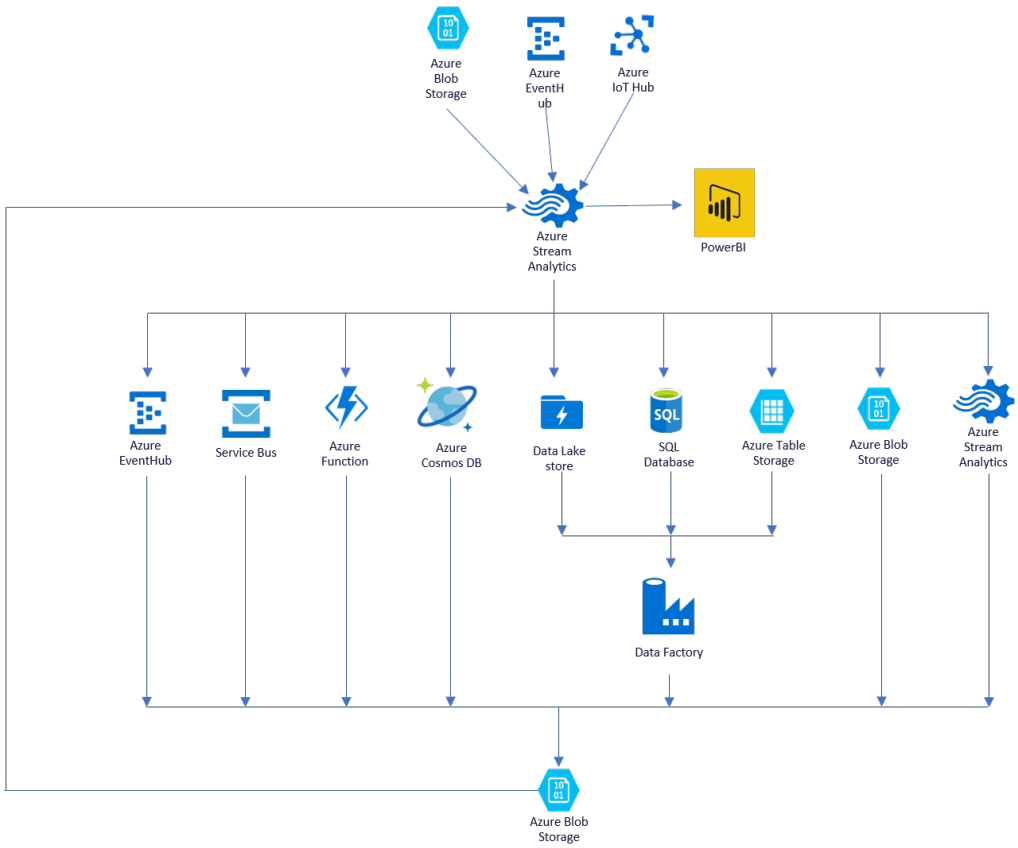

Figura 4.5: Esquema da monitorização da qualidade dos dados em grande escala (Elaboração própria)

4.1.3 Resposta da API

Para fazer uma monitorização da resposta da API foi implementado uma *framework* que tem como objetivo chamar uma API e receber a resposta da mesma. Para isso foi criado uma função que simula o comportamento de uma API. Esta simulação da API verifica se recebeu a informação

proveniente da chamada da API, retornando um *Status Code OK* no caso de sucesso e um *Status code BadRequest* no caso contrário.

Para monitorizar a resposta da API foi implementada uma função em C# na *Function App*, que chama API e recebe a sua resposta usando um método de requisição de HTTP, chamado POST. No caso de insucesso da chamada de API, a função responsável pela monitorização vai voltar a chamar até 3 vezes. Assim é garantido que mesmo que exista um erro, ou uma má conexão, o sistema vai tentar 3 vezes antes de o erro ser reportado. De seguida, a resposta é guardada juntamente com o número de tentativas e a data numa pasta da *blob storage*, figura 4.6.

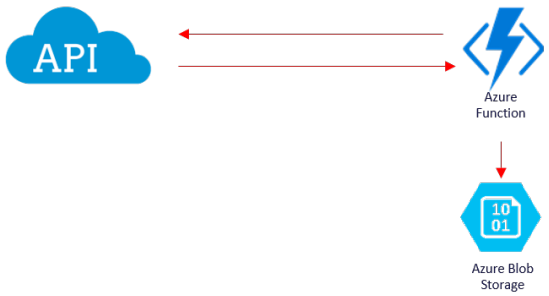

Figura 4.6: Esquema da monotorização da API até à *Blob storage* (Elaboração própria)

De seguida, para que esta monitorização seja visualizada no *Power BI*, a *Blob* vai ser uma entrada em *stream* do *stream analytics* e o output vai ser o *Power BI* por meio de uma *query*. Desta forma é possível ver no *Power BI* quantas chamadas de API foram bem sucedidas, quantas não foram, quantas tentativas até a chamada da API ter sucesso e até as horas a que existiram os insucessos, tudo em tempo real. Na figura 4.7 encontra-se o esquema completo da monitorização da resposta da API.

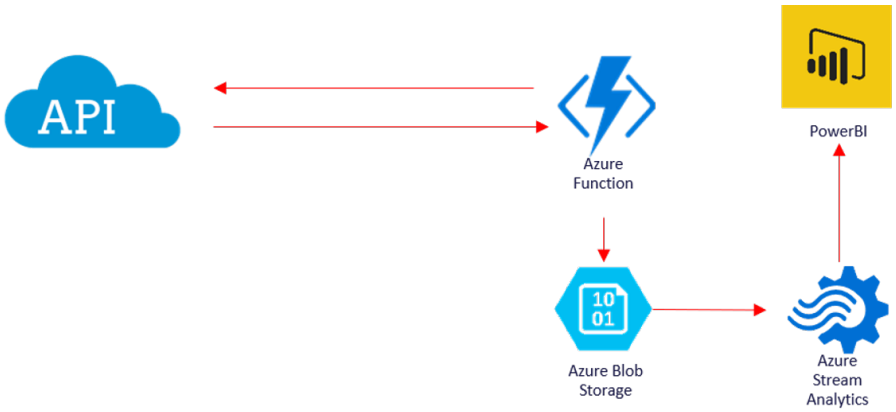

Figura 4.7: Esquema da monitorização da resposta da API (Elaboração própria)

4.1.4 *Logs* de diagnóstico de recursos

Com o objetivo de descobrir se os recursos estão a funcionar corretamente foi criada uma monitorização para os *logs* de diagnóstico dos recursos. Como referido no capítulo 3, só é possível usar os *logs* de diagnóstico do *Event Hubs* e do *Stream Analytics*. Em relação à *Azure Function* e à *Blob storage* é possível também fazer uma monitorização de logs, pois estes são guardados de forma automática numa *Table Storage*. A questão é que o *Stream analytics* não provém de uma entrada para a *Table Storage*, por isso a informação teria de ser, por exemplo, copiada para uma *blob* antes de passar para o *Stream analytics* ou mesmo criar uma função que fosse buscar os valores da *Table Storage* e os enviasse para o *Event Hubs* e de seguida para o *Stream analytics*. Contudo, esses *logs* podem ser consultados manualmente na *Table Storage*.

Para usar os *logs* do *Event Hubs* é necessário configurar uma funcionalidade do *Event Hubs* chamada *diagnostic logs* onde é possível passar os *logs* desejados tanto para uma pasta da *blob* como para um *Event Hubs*. Nesta *framework* os *logs* são guardados numa *blob* e transmitidos também para um *Event Hubs*. A *blob* tem como objetivo guardar os dados como histórico e o *Event Hubs* passa esses mesmos *logs* para o *Stream analytics* onde vão ser analisados para mais tarde serem expostos no *Power BI*, como podemos verificar no esquema da solução na figura 4.8.

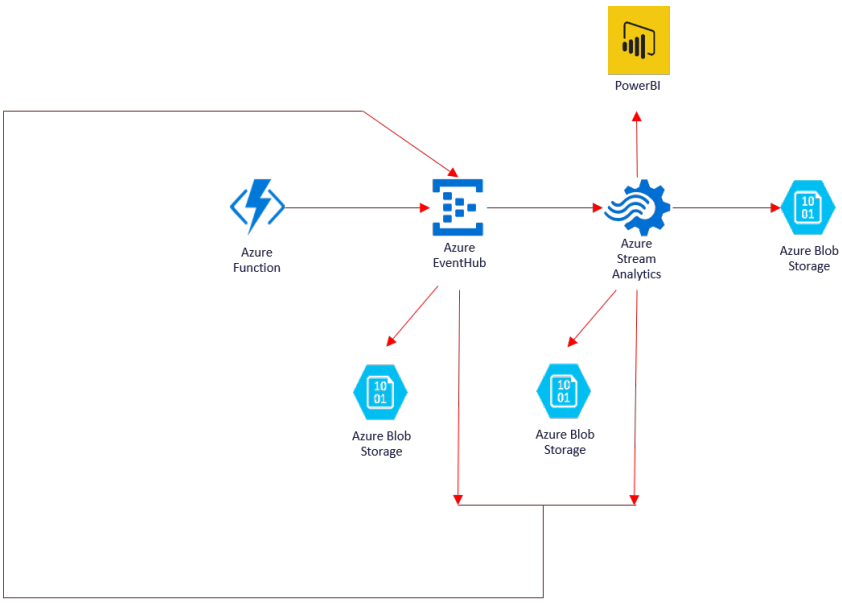

Figura 4.8: Esquema da monitorização dos *Logs* dos recursos (Elaboração própria)

Capítulo 5

Resultados

Neste capítulos são abordados dois resultados da *framework* implementada. Em primeiro lugar, é possível visualizar os resultados no Power BI e o seu significado. De seguida, foi feita uma análise orçamental, onde se percebe o custo total da *framework*.

5.1 Power BI

Neste capítulo é possível visualizar os resultados da monitorização realizada no âmbito desta dissertação. Para isso as monitorizações são visualizadas separadamente para que a compreensão destas seja mais fácil. Desta forma a figura 5.1, mostra a monitorização da qualidade de dados realizada, descrita no subcapítulo 3.3.2. Este relatório em específico mostra se os dados se perdem a passar da *Azure function* para o *Event Hubs*. Assim, observa-se o total de transações completas recebidas da parte do cliente na *Azure Function*. É possível ver o total de sucessos e de insucessos, ou seja o número de transações que chegam ao *Event Hubs* e o número das que não chegam, respetivamente. O *Power BI* provém de filtros iterativos, que possibilita filtrar a informação presente no relatório por dia, por hora e por país. Podendo assim oferecer uma experiência ao utilizador mais interativa e interessante. Também é possível ver numa tabela se o *messageID* corresponde a sucesso ou a insucesso e a data respetiva.

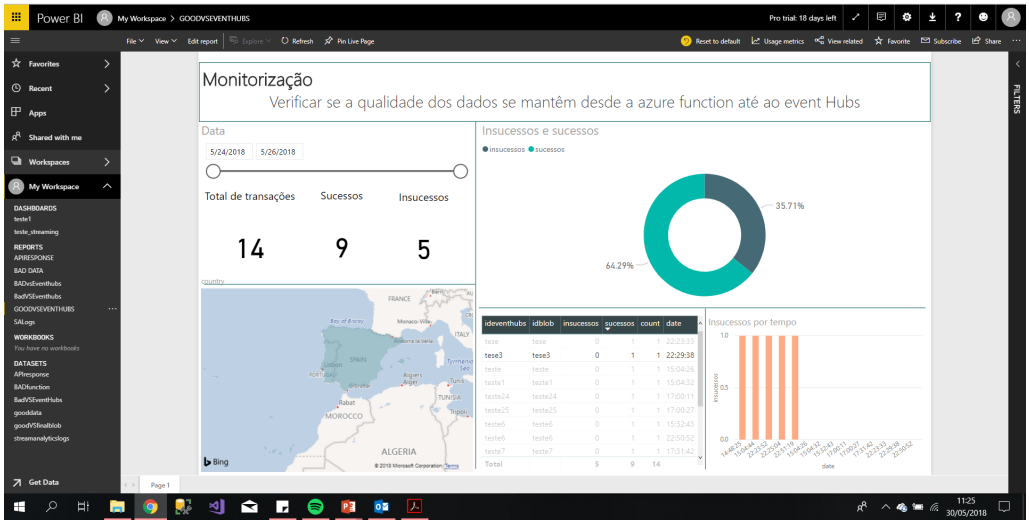

Figura 5.1: Relatório no *Power BI* de qualidade dos dados da *Azure function* para o *Event Hubs*

Na figura 5.2 é possível visualizar as transações que chegam à *Azure function* corrompidas, ou seja, que contêm campos em falta. Assim, é visualizado no *Power BI* um total de campos danificados e o número total de campos danificados na mesma transação. Também é possível ver a percentagem de campos danificados, assim é verificada a existência de padrões, como por exemplo um campo que vem danificado de forma recorrente. Foi construído um gráfico onde pode ser visto quantos campos foram corrompidos de acordo com as horas. É possível igualmente aceder a uma lista que dá o *message ID* da transação e a contagem de campos danificados na mesma que pode ser filtrada de acordo com a data.

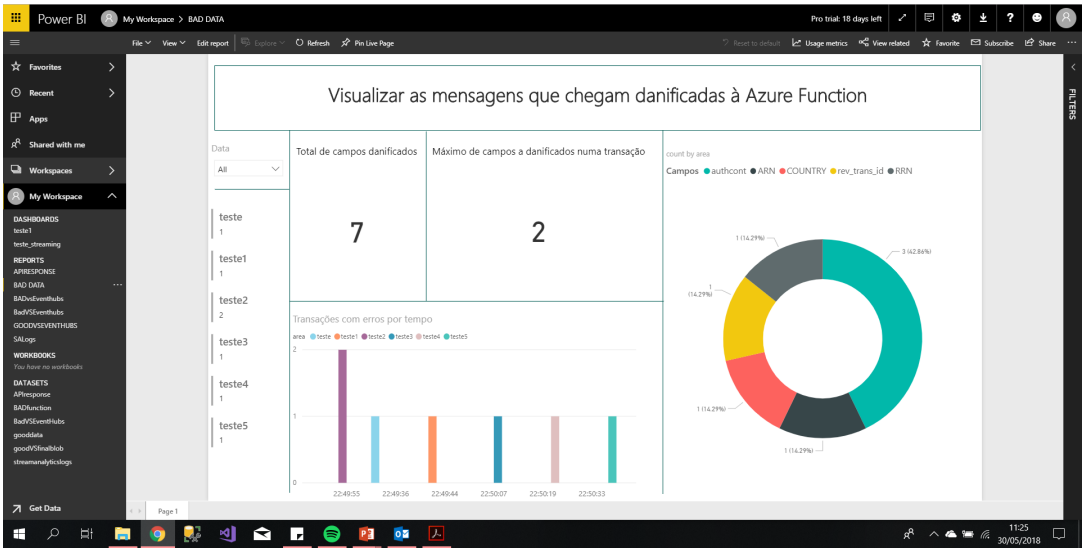

Figura 5.2: Relatório no *Power BI* de qualidade dos dados das transações danificadas no ponto inicial

Para que fosse possível verificar a taxa de sucesso do projeto em questão foi feita uma monitorização que compara as transações que chegam do início do processo até ao final, como se pode observar na figura 5.3. Assim, é observado o número de transações iniciais, o número de transações finais e o número de insucessos segundo esses valores. É apresentada uma tabela onde indica o *message ID* da mensagem que chega ao ponto inicial do projeto e o *message ID* da transação que chega ao ponto final do projeto. Quando o campo correspondente às transações finais não estiver preenchido significa que a transação em questão não chegou ao ponto final do processo. É possível também neste relatório filtrar a informação segundo a data e a localização geográfica.

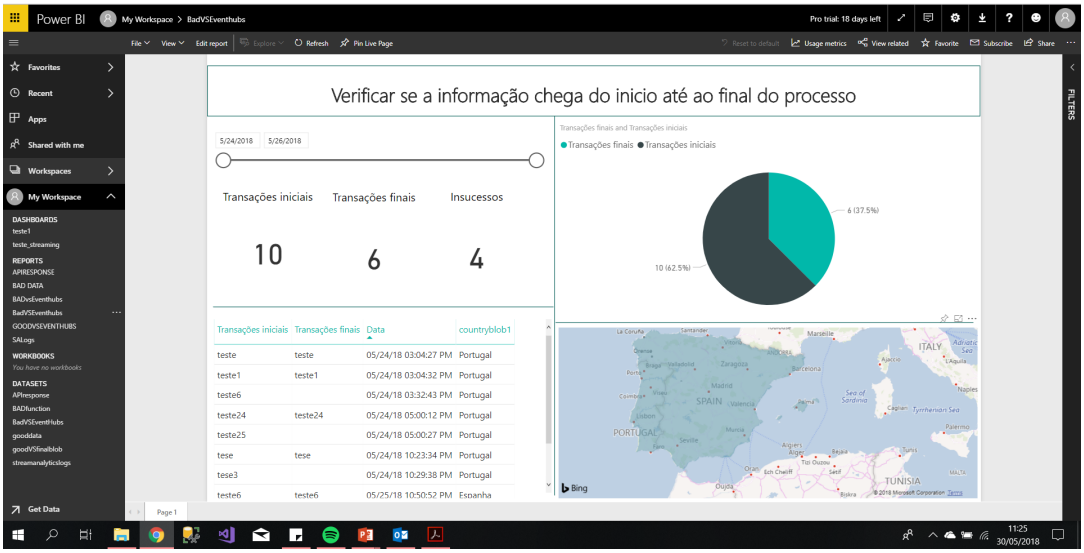

Figura 5.3: Relatório no *Power BI* de qualidade dos dados das transações iniciais comparativamente com as finais

Para perceber o comportamento da API foi realizada uma monitorização das respostas da API, como descrito no subcapitulo 3.3.3. Com esse objetivo, na figura 5.4 vê-se as respostas da API do processo. É possível ver quantas respostas existiram com *Status Code OK*, *Status code BadRequest* e também *Status Code Internal Server Error*. Estas que foram as repostas mais comuns ao longo da construção do processo. É possível ver um gráfico onde as respostas são comparadas em percentagem, neste momento é visível que existe 98,72% de respostas com *Status Code OK*. Também é possível visualizar uma tabela onde se consegue ver o *Status Code*, a data do mesmo e o número de tentativas necessárias até ao sucesso.

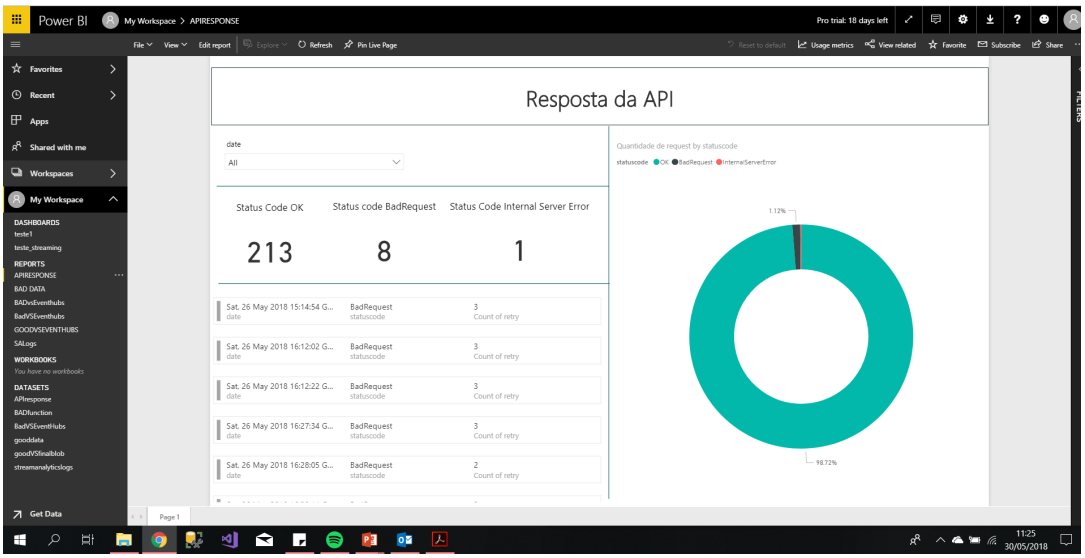

Figura 5.4: Relatório no *Power BI* da resposta da API

Para a monitorizar os *logs* dos recursos foi feito um relatório no *Power BI*, figura 5.5, onde é possível ver os *logs* do *Stream analytcs*. É visualizada a quantidade de erros que ocorreram no *Stream Analytics*, o total de *logs* existentes e os *logs* informacionais. Existe uma tabela onde é possível verificar os erros e onde ocorreram, outra onde estão presentes os *logs* da categoria *execution* e noutra tabela os *logs* de *authoring*. Os *logs* do *Event Hubs* deviam ser apresentados dentro dos mesmos padrões, mas não foi possível criar um relatório pois para estes, pois para isso seria preciso apagar, alterar ou criar *Event Hubs* o que não foi possível no âmbito deste processo.

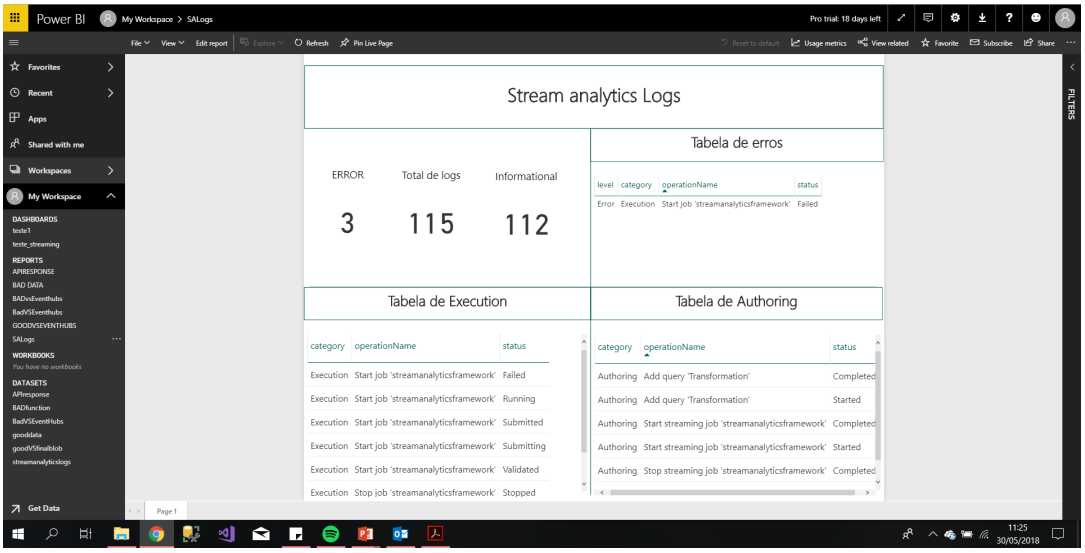

Figura 5.5: Relatório no *Power BI* dos *logs* de diagnóstico

5.2 Orçamento

Para o cálculo do orçamento final da *framework* foi feita uma análise do preço de cada recurso separadamente no capítulo 3.1.6. Assim, neste capítulo foi feito o orçamento primeiramente para a simulação de alerta de fraudes e de seguida o orçamento final para se perceber o aumento nos custos com a *framework* de monitorização inserida. Este orçamento foi feito apenas para um mês de uso do projeto. Foi admitido que são recebidos 4 eventos por segundo na *Azure Function*, que o projeto funciona 24h por dia e que cada mês são 31 dias.

5.2.1 Orçamento para a simulação de alerta de fraudes

Neste subcapítulo vamos fazer uma análise orçamental para a simulação de alerta de fraudes. Ou seja, sem a *framework* de monitorização.

5.2.1.1 Azure Function

Para calcular o preço da *Azure Function* é preciso calcular o consumo dos recursos (CR) e as Execuções (EX) desta.

$$P = CR + EX$$

Na *Azure function* são recebidos:

$$4 eventos * 60s * 60min * 24h * 31d = 10\,713\,600 \text{ eventos/mês}$$

É considerado que se recebe 11 milhões de eventos por mês. A função utiliza 500 MB (que segundo a norma arredonda para os 128 MB) por execução e esta é executada 11 milhões de vezes por mês e demora 500 ms por cada execução.

Foi calculado em primeiro lugar o consumo por recurso:

$$CR = 11M * 0,5s * 512/1024GB = 2,8MGB/s$$

Como é oferecido por mês 400.000 GB então:

$$CR = 2.8M - 400.000 = 2.4MGB/s$$

Relativamente ao preço das execuções vai ser, considerando que a Microsoft oferece 1 milhão de eventos por mês:

$$EX = 11M - 1M = 10M$$

Concluindo o preço final vai ser:

$$P = 2.4M * 0,000014€ + 10 * 0,169€ = 35,29€$$

5.2.1.2 *Event Hubs*

No caso do *Event Hubs* foi escolhida a *tier standard* pois a básica não permite ter mais do que um *consumer group*. Como cada *consumer group* só suporta até 5 consumidores foi sentida a necessidade de ter mais do que um. Por essa razão os preços calculados para o *Event Hubs* são de acordo com a *tier Standard*.

É sabido que o *Event Hubs* recebe aproximadamente 11 milhões de eventos por mês. Este recurso cobra por Eventos de Entrada (EE) e Unidades de débito (UD).

$$P = EE + UD$$

Assim relativamente aos eventos de entrada (EE):

$$EE = 0,024€ * 11 = 0,264€/\text{mês}$$

Agora para calcular o preço por Unidade de débito foi considerado que cada evento que o *Event Hubs* recebe tem aproximadamente 0,0001 MB. Por segundo o *Event Hubs* recebe então 0,0004 MB. Valor que corresponde a uma unidade débito pois é inferior a 1 MB/s. Ou seja, por mês o preço da unidade de débito é:

$$UD = 0,026€ * 24h * 31d = 19,34€.$$

Em suma, o preço final referente a este recurso é:

$$P = 19,34€ + 0,264€ = 19,608€/\text{mês}$$

5.2.1.3 *Stream Analytics*

O *Stream Analytics* cobra pela quantidade de *Streaming Units* (SU) usadas no recurso. *Streaming Unit* é a capacidade de computação do evento em relação ao CPU, memória e taxas de leitura e escrita.

Assim, para este projeto escolheu-se usar 3 *Stream Units* e à medida que o projeto foi construído as *Stream Units* foram alteradas de modo a responder as necessidades.

Como foram usadas 3 SU o preço do *Stream Analytics* é:

$$P = 0,102€ * 3SU * 24h * 31d = 227,664€/\text{mês}$$

5.2.1.4 *Blob Storage*

Para calcular o preço da *Blob Storage* é preciso dividir em duas partes diferentes. É necessário calcular o preço de armazenamento de dados (AD) e o preço das operações e transferências de dados na *Blob* (OTD). Assim, em relação aos preços de armazenamento de dados: São armazenados por mês

$$Tamanho = 4e * 0,0001MB * 60s * 60min * 24h * 31d = 1071,36\text{MB/mês}$$

O que corresponde a menos de 50 TB/mês logo o preço por armazenamento de dados por mês é:

$$AD = 0,01666€ * 104,625GB = 1,743€/\text{mês}$$

Falando agora de preços de operações e transferência de dados: Por mês foram escritas 11 milhões de eventos na blob final do processo. Neste projeto como já foi referenciado em cima foram usadas *blob* com acesso frequente. O que significa que o preço relativo às operações e transferências de dados refere-se simplesmente às operações de escrita (OE).

$$OE = 11M/10.000 * 0,05€ = 55€$$

Concluindo o preço final por mês do uso da *Blob Storage* é:

$$P = 1,743€ + 55€ = 56,74€/\text{mês}$$

5.2.1.5 *Preço final*

Na tabela 5.1 é possível ver o preço por recurso e seguidamente o preço final para o processo de simulação de alerta de fraudes.

Tabela 5.1: Preço final da simulação do caso de estudo da alerta de fraudes

Recursos	**Preço** (€)
Azure Function	35,12 €
Event Hubs	19,60 €
Stream analytics	227,66 €
Blob Storage	56,74 €
Total	339,12 €

5.2.2 Orçamento final

Neste subcapítulo foi então feito orçamento final do projeto. Este inclui o processo de simulação de deteção de fraudes com a adição da *framework* de monitorização.

5.2.2.1 Azure Function

O preço referente a *Azure Function* mantem-se igual ao do subcapítulo 5.2.1.1. Pois este continua a ter o mesmo tempo de execução e os mesmos eventos de entrada com o mesmo tamanho.

5.2.2.2 Event Hubs

Com a adição da framework de monitorização, vão agora existir 4 Event Hubs diferentes

- *Event Hubs* referente à simulação da deteção de fraudes que recebe aproximadamente 4 transações por segundo;

- *Event Hubs* referente aos *Logs* do *Stream Analytics (Authoring)* admitindo que recebe aproximadamente 2 transações por hora;

- *Event Hubs* referente aos *Logs* do *Stream Analytics (Execution)* admitindo que recebe aproximadamente 2 transações por hora;

- *Event Hubs* referente aos *Logs* do *EventHubs (Archive)* admitindo que recebe aproximadamente 2 transações por hora;

- *Event Hubs* referente aos *Logs* do *Event Hubs (Operational)* admitindo que recebe aproximadamente 2 transações por hora;

Sabendo que 2 eventos por hora são:

$$2e * 24h * 31d = 1.488\text{e/mês}$$

Ou seja, menos de 1 milhão. É considerado também que cada mensagem de *log* tem aproximadamente 0,235 MB. Como os *Event Hubs* recebem 2 transações por hora, então não vai ser necessário mais do que uma unidade de débito, pois estes recebem menos de 1 MB/s de entrada. Assim sendo para os diferentes *Event Hubs* temos os preços representados na tabela 5.2.

Tabela 5.2: Preço final do *Event Hubs* da *framework* de monitorização

Event Hubs	Preço por mês (€)
Event Hub referente à simulação da deteção de fraudes	Eventos de Entrada=0,024€*11 =0,264 €
	Unidade de débito =19,34 €
Event Hub referente aos Logs do Stream analytics (Authoring)	Eventos de Entrada =0,024€
	Unidade de débito =19,34 €
Event Hub referente aos Logs do Stream analytics (Execution)	Eventos de Entrada =0,024€
	Unidade de débito =19,34 €
Event Hub referente aos Logs do EventHubs (Archive)	Eventos de Entrada =0,024€
	Unidade de débito = 19,34 €
Event Hub referente aos Logs do Event Hubs (Operational)	Eventos de Entrada =0,024€
	Unidade de débito = 19,34 €

Então o preço final referente ao *Event Hubs* vai ser então 97,06 euros por mês.

5.2.2.3 *Stream Analytics*

Com o aumento de entradas no *Stream analytics* foi sentida a necessidade de aumentar o número de *Streaming units*, para alcançar uma melhor performance. Assim, foi alterado para 6 SU, alterando assim o preço do *Stream analytics*:

$$P = 0,102€ * 6SU * 24h * 31d = 455,328€/\text{mês}$$

5.2.2.4 *Blob Storage*

Para o cálculo dos custos da *Blob Storage* é preciso fazer o orçamento para 7 *containers* na *Blob Storage* diferentes:

- Blob final
- Blob com dados completos

- Blob com dados incompletos
- Blob logs de diagnóstico do stream analytics (Authoting)
- Blob logs de diagnóstico do stream analytics (Execution)
- Blob logs de diagnóstico do Event Hubs (Archive)
- Blob logs de diagnóstico do Event Hubs (Operational)

Para a calcular o preço relativamente ao armazenamento de dados foi calculado o espaço ocupado por cada *Container* da *blob*, como se ver na tabela 5.3. Para isso foi admitido que 10% das transações que chegam ao *Event Hubs* estão danificadas.

Tabela 5.3: Preço da *Blob Storage* da *framework* de monitorização relativamente ao armazenamento de dados

Blob	**Tamanho de armazenamento por mês (MB)**
Blob final	4e*0,0001 MB* 60s*60min*24h*31d=1071,36 MB
Blob com dados completos	90%*4e*0,0001MB*60s* 60min*24h*31d=964,224 MB
Blob com dados incompletos	10%*4e/s*0,0001MB*60s*60min*24h*31d=107,136 MB
Blob logs de diagnóstico do stream analytics (Authoring)	2e/h*0,235MB*24h*31d=349,68 MB
Blob logs de diagnóstico do stream analytics (Execution)	2e/h*0,235MB*24h*31d=349,68 MB
Blob logs de diagnóstico do Event Hubs (Archive)	2e/h*0,235MB*24h*31d=349,68 MB
Blob logs de diagnóstico do Event Hubs (Operational)	2e/h*0,235MB*24h*31d=349,68 MB
Total	3541,44 MB=0,00354144 TB

Ou seja o preço de armazenamento de dados (AD) vai ser:

$$AD = 0,01666€ * 3541,44GB = 59€/\text{mês}$$

Para calcular o preço de operações e transferências de dados foi calculado quantas operações de escrita foram efetuadas em cada *container* da *blob*, como se pode verificar na tabela 5.4.

Tabela 5.4: Preço da *Blob Storage* da *framework* de monitorização relativamente às operações

Blob	**Nº de operações de escrita por mês**
Blob final	11 M
Blob com dados completos	90%*11M=9,9M
Blob com dados incompletos	10%*11M=1.1 M
Blob logs de diagnóstico do stream analytics (Authoring)	2e* 24h*31d=1488
Blob logs de diagnóstico do stream analytics (Execution)	2e* 24h*31d=1488
Blob logs de diagnóstico do Event Hubs (Archive)	2e* 24h*31d=1488
Blob logs de diagnóstico do Event Hubs (Operational)	2e* 24h*31d=1488
Total	22005952 aproximadamente 22 milhões

Em relação a preços de operações e transferência de dados(OTD):

$$OTD = 22.005.952/10.000 * 0,05€ = 110,0298€/\text{mês}$$

Concluindo o preço final por mês do uso da Blob Storage é:

$$P = 59€ + 110,0298€ = 169€/\text{mês}$$

5.2.2.5 *Preço final*

Na tabela 5.1 é possível ver o preço por recurso e seguidamente o preço final para o processo de simulação de alerta de fraudes com a *framework* de monitorização.

Tabela 5.5: Preço final da *framework* de monitorização

Recursos	**Preço** (€)
Azure Function	35,12 €
Event Hubs	97,06 €
Stream analytics	455,32 €
Blob Storage	169,00 €
Total	756,60 €

Capítulo 6

Conclusões e Trabalho Futuro

No presente capítulo serão apresentadas as conclusões alcançadas durante a implementação do projeto desenvolvido. Além disso, serão mencionadas possíveis melhorias ao sistema atual, como perspetivas futuras.

6.1 Satisfação dos Objetivos

Neste projeto foi desenhada e implementada uma *framework* de monitorização capaz de supervisionar qualquer processo que envolva os recursos abordados no âmbito da dissertação.

Esta *framework* foi desenhada após um estudo, apresentado na revisão bibliográfica, sobre *Big Data* e ferramentas de *Cloud Computing*. Foi também evidenciada a necessidade do uso de tecnologias inovadoras aquando da abordagem ao tema Big Data. Com o mesmo objetivo foi realizado um caso de estudo para a empresa, facilitando assim a deteção de possíveis falhas que pudessem ocorrer no processo de forma mais fundamentada e realista.

Na implementação da *framework* de monitorização foram experimentadas várias hipóteses com o intuito de perceber qual o desenho mais adequada para a mesma. Avaliando aspetos como a eficiência e o custo.

Concluindo foi então implementada uma *framework* de monitorização que cumpre os requisitos iniciais estabelecidos. O processo é monitorizado em vários pontos, onde é observado a qualidade dos dados, as respostas das APIs e os *logs* de diagnóstico. É possível também visualizar a monitorização realizada em relatórios do *Power BI*. Aqui são feitas várias métricas como por exemplo, a contagem de transações iniciais e o estado de chegada.

O desenvolvimento deste projeto possibilitou a aquisição de competências técnicas relativamente à familiarização com as ferramentas utilizadas (*Microsoft Azure* e *Microsoft Power BI*) e com conceitos inerentes ao tema *Big Data*. Para além disso, a realização desta dissertação em contexto empresarial proporcionou uma experiência profissional, tendo assim a oportunidade de realizar uma análise mais ampla e direcionada em relação ao problema proposto.

6.2 Trabalho futuro

Como trabalho futuro, sugeria que na presença de uma falha no processo fosse possível relançar processos de uma forma automática, pensando nas dependências do mesmo. Assim, para além de monitorização, também ia existir um processo paralelo capaz de recuperar e reenviar informação a partir do ponto onde existiu a falha.

Anexo A

Código de Exemplo de uma transação

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/
soap/envelope/">
<soap:Body>
<ns3:notify xmlns:ns2="http://www.mastercard.com/wsdl"
xmlns:ns3="http://api.wsclient.
notificationmodule.mastercard.com/">
<ns2:NotificationMessage>
<messageID>58daf84e-da70-4717-a189-b2ec74f43f85</messageID>
<fields>
<key>AUTH_ID</key>
<value>103325</value>
</fields>
<fields>
<key>REV_TRANS_ID</key>
<value>104075</value>
</fields>
<fields>
<key>ARN</key>
<value></value>
</fields>
<fields>
<key>RRN</key>
<value>080000100001</value>
</fields>
<fields>
<key>TRANS_CODE</key>
<value>T.A1.T</value>
</fields>
```

```
<fields>
<key>DECL_REASON_CODE</key>
<value>0</value>
</fields>
<fields>
<key>COUNTRY</key>
<value>United States</value>
<fields>
</fields>
<key>TRANS_DATE</key>
<value>2018-01-18 13:42:42</value>
</fields>
</ns2:NotificationMessage>
</ns3:notify>
</soap:Body>
</soap:Envelope>
```

.

Anexo B

Dicionário de Respostas HTTP mais frequentes

- **100 Continue**

  É enviado em certas circunstâncias, quando um cliente envia uma solicitação contendo um corpo. A resposta indica que os cabeçalhos de solicitação foram recebidos e que o cliente deve continuar a enviar o corpo. O servidor retorna uma segunda resposta quando a solicitação foi concluída.

- **200 OK**

  Indica que o pedido foi bem sucedida e que a resposta do *body* tem o resultado do pedido.

- **201 Created**

  É devolvida uma resposta a um pedido PUT para indicar que a este foi bem-sucedido.

- **301 Moved Permanently**

  Redireciona o utilizador permanentemente para um URL diferente, que é especificada no cabeçalho. O cliente deve usar o novo URL no futuro, em vez do original.

- **302 Found**

  Redireciona o utilizador de forma temporária para um URL diferente, que é especificado no cabeçalho. O utilizador deve reencaminhar para o URL original nos pedidos seguintes seguintes.

- **400 Bad Request**

  Indica que o cliente apresentou um pedido HTTP inválido.

- **401 Unauthorized**

  Indica que o servidor requer autenticação HTTP antes do pedido ser aceite.

- **404 Not Found**

  Indica que o recurso pedido não existe.

- **500 Internal Server Error**

  Indica que o servidor encontrou um erro ao tentar responder ao pedido. Isso acontece normalmente quando é apresentado um input inesperado que causa um erro em alguma parte do processamento da API.

Referências

[1] https://azure.microsoft.com/pt-pt/overview/what-is-iaas/, cited July 2018.

[2] https://azure.microsoft.com/pt-pt/overview/what-is-paas/, cited July 2018.

[3] https://azure.microsoft.com/pt-pt/overview/what-is-saas/, cited July 2018.

[4] https://www.theregister.co.uk/2017/06/19/gartner_confirms_what_we_all_know_aws_and_microsoft_are_the_cloud_leaders_by_a_fair_way/, cited July 2018.

[5] https://medium.com/@metse/using-azure-functions-4d989945efa3, cited July 2018.

[6] https://thirdeyedata.io/azure-event-hub/, cited July 2018.

[7] https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction, cited July 2018.

[8] https://docs.microsoft.com/pt-pt/azure/data-factory/copy-activity-overview, cited July 2018.

[9] https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-of-diagnostic-logs, cited July 2018.

[10] H.P. Luhn. A Business Intelligence System. *IBM Journal of Research and Development*, 2(4):314–319, 1958. URL: http://altaplana.com/ibmrd0204H.pdf, doi:10.1147/rd.24.0314.

[11] Manning Publications For e Xiang Zhong. MEAP Edition Manning Early Access Program Big Data Version 15.

[12] Daniel J Power. A brief history of decision support systems. *http://DSSResources.COM/history/dsshistory.html*, 4(January 2007):1–17, 2007. URL: http://dssresources.com/history/dsshistory.html.

[13] Steve Lavalle, Eric Lesser, Rebecca Shockley, Michael S Hopkins, e Nina Kruschwitz. Big Data, Analytics and the Path From Insights to Value. *MIT Sloan Management Review*, 52(2):21–32, 2011. doi:10.0000/PMID57750728.

[14] Carlos Sezões, José Oliveira, e Miguel Baptista. *Business Intelligence*. 2006. URL: http://web.spi.pt/negocio{_}electronico/documentos/manuais{_}PDF/Manual{_}V.pdf, doi:10.1002/9780470753866.

[15] Steve Williams e Nancy Williams. *Critical Acclaim for The Profit Impact of Business Intelligence*. 2007. arXiv:arXiv:1011.1669v3, doi:10.1017/CBO9781107415324.004.

[16] Ralph Kimball e Margy Ross. *The data warehouse toolkit: the complete guide to dimensional modelling*. 2011. URL: http://190112.8m.com/Bibliografia.pdf, arXiv:arXiv:1011.1669v3, doi:10.1145/945721.945741.

[17] W.H. Inmon. *Building the data warehouse*. 2002.

[18] Daniel Moody e Mark A.R Kortink. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000)*, 2000:5–16, 2000.

[19] David Loshin. *Business Intelligence : The Savvy Manager's Guide, Getting Onboard With Emerging IT*. 2003. arXiv:arXiv:1011.1669v3, doi:10.1017/CBO9781107415324.004.

[20] A Dimensional Modeling Manifesto by ralph kimball. https://www.kimballgroup.com/1997/08/a-dimensional-modeling-manifesto/. Accessed: 2018-02-15.

[21] João Ferreira, Miguel Miranda, António Abelha, e José Machado. O Processo ETL em Sistemas Data Warehouse. *INForum 2010 - II Simpósio de Informática*, páginas 757–765, 2010. arXiv:2.

[22] ETL.pdf.

[23] Microsoft Windows e Mac Os. *Data Warehousing in the Age of Big Data*, volume XXXIII. 2014. arXiv:arXiv:1011.1669v3, doi:10.1007/s13398-014-0173-7.2.

[24] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, e Angela H. Byers. Big data: The next frontier for innovation, competition, and productivity, Maio 2011. URL: http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation.

[25] MS Windows NT kernel description. file:///C:/Users/inesv/Documents/OneDrive/Tese/Estado%20de%20arte/Bigdata/(The%20Morgan%20Kaufmann%20series%20on%20business%20intelligence)%20Krish%20Krishnan-Data%20warehousing%20in%20the%20age%20of%20big%20data-Elsevier%20_%20Morgan%20Kaufmann%20(2013).epub. Accessed: 2018-03-16.

[26] An Attivio e White Paper. Big Data & New Business Value Creation Powered by Unified Information Access.

[27] Bhashyam Ramesh. *Big Data Architecture*. 2015. arXiv:arXiv:1011.1669v3, doi:10.1145/1815961.1815963.

[28] Santhosh Baboo e P. Renjith Kumar. Next Generation Data Warehouse Design with Big Data for Big Analytics and Better Insights. *Global Journal of Computer Science and Technology*, 13(7):19–23, 2013. URL: http://computerresearch.org/stpr/index.php/gjcst/article/viewArticle/1473.

[29] Paul Zikopoulos e Chris Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data: Analytics for Enterprise Class Hadoop and Streaming Data*. 2011.

[30] The Age of Big Data By STEVE LOHR. http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html. Accessed: 2018-03-16.

[31] Yuri Demchenko, Paola Grosso, Peter Membrey, Cees De Laat, e Peter Membrey. Addressing big data issues in Scientific Data Infrastructure. *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, páginas 48–55, 2013. doi:10.1109/CTS.2013.6567203.

[32] Sam Madden. to Big Data. páginas 0–2, 2012.

[33] Robert Heinrich, Reiner Jung, Christian Zirkelbach, Wilhelm Hasselbring, e Ralf Reussner. *Software Architecture for Big Data and the Cloud*. 2017.

[34] Pavan Sridhar e Neha Dharmaji. A Comparative Study on How Big Data is Scaling Business Intelligence and Analytics. *International Journal of Enhanced Research in Science Technology & Engineering*, 2(8):2319–746387, 2013.

[35] Cecília Almeida Rodrigues Lima Janaina de Holanda Costa Calazans. Performances Interacionais e Mediações Sociotécnicas PEGADAS DIGITAIS: "BIG DATA "E INFORMAÇÃO ESTRATÉGICA SOBRE O CONSUMIDOR 1. 2013.

[36] Ounacer Soumaya, Talhaoui Mohamed Amine, Ardchir Soufiane, Daif Abderrahmane, e Azouazi Mohamed. Real-time Data Stream Processing Challenges and Perspectives. 14(5):6–12, 2017.

[37] Wolfram Wingerath, Felix Gessert, Steffen Friedrich, e Norbert Ritter. Real-time stream processing for Big Data. *it - Information Technology*, 58(4):186–194, 2016. URL: https://www.degruyter.com/view/j/itit.2016.58.issue-4/itit-2016-0002/itit-2016-0002.xml, doi:10.1515/itit-2016-0002.

[38] M Barlow. *Real-Time Big Data Analytics: Emerging Architecture*. 2013. URL: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:No+Title{#}0{%}5Cnhttp://books.google.com/books?hl=en{&}lr={&}id=64Uba0n38R4C{&}oi=fnd{&}pg=PP2{&}dq=Real-Time+Big+Data+Analytics:+Emerging+Architecture{&}ots=vY6OSTRPt6{&}sig=SM9ublNwvU87Nv26994gF8465No, arXiv:arXiv:1011.1669v3, doi:10.1007/s13398-014-0173-7.2.

[39] Nathan Marz e James Warren. *Big Data, Principles and best practices of scalable real-time data systems*, volume 37. Manning Publications, 2015. arXiv:1-933988-16-9, doi:10.1073/pnas.0703993104.

[40] Questioning the Lambda Architecture by jay kreps. https://www.oreilly.com/ideas/questioning-the-lambda-architecture. Accessed: 2018-02-21.

[41] Kappa architecture. 2010. URL: http://events.linuxfoundation.org/sites/events/files/slides/ASPgems-KappaArchitecture.pdf.

[42] D Agrawal, S Das, e Amr El Abbadi. Big data and cloud computing: new wine or just new bottles? *Proceedings of the VLDB Endowment*, 3(1-2):1647–1648, 2010. URL: http://dl.acm.org/citation.cfm?id=1921063, doi:10.14778/1920841.1921063.

[43] Marcello Trovati e Richard Hill. *Book – Big-Data Analytics and Cloud Computing*. 2015. URL: http://dblp.uni-trier.de/db/books/collections/THAZL2015.html, doi:10.1007/978-3-319-25313-8.

[44] M Armbrust, A Fox, R Griffith, AD Joseph, e RH. Above the clouds: A Berkeley view of cloud computing. *University of California, Berkeley, Tech. Rep. UCB*, páginas 07–013, 2009. URL: http://scholar.google.com/scholar?q=intitle:Above+the+clouds:+A+Berkeley+view+of+cloud+computing{#}0, arXiv:0521865719 9780521865715, doi:10.1145/1721654.1721672.

[45] Michael Armbrust, Ion Stoica, Matei Zaharia, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, e Ariel Rabkin. A view of cloud computing. *Communications of the ACM*, 53(4):50, 2010. URL: http://portal.acm.org/citation.cfm?doid=1721654.1721672, arXiv:0521865719 9780521865715, doi:10.1145/1721654.1721672.

[46] Pedro Caldeira Neves, Bradley Schmerl, Javier Cámara, e Jorge Bernardino. Big Data in Cloud Computing: Features and Issues. *Proceedings of the International Conference on Internet of Things and Big Data*, (October):307–314, 2016. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005846303070314, doi:10.5220/0005846303070314.

[47] Business Intelligence. *No Title*.

[48] Peter Mell e Timothy Grance. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *National Institute of Standards and Technology, Information Technology Laboratory*, 145:7, 2011. URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, arXiv:2305-0543, doi:10.1136/emj.2010.096966.

[49] H Venkatesh, Shrivatsa D Perur, e Nivedita Jalihal. A Study on Use of Big Data in Cloud Computing Environment. 6(3):2076–2078, 2015.

[50] Peter Heller e Dee Piziak. An Enterprise Architect ' s Guide to Big Data. *Oracle Enterprise Architecture White Paper*, (February):4–15, 2015. URL: http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf.